On Discontinuous Galerkin Methods for Singularly Perturbed and Incompressible Miscible Displacement Problems

CHAPMAN, JOHN,ROBERT

On Discontinuous Galerkin Methods for Singularly Perturbed and Incompressible Miscible Displacement Problems

John Chapman

A Thesis presented for the degree of
Doctor of Philosophy

Numerical Analysis Group
Department of Mathematical Sciences
University of Durham
England

November 2012

Dedicated to

Laura, of course.

On Discontinuous Galerkin Methods for Singularly Perturbed and Incompressible Miscible Displacement Problems

John Chapman

Submitted for the degree of Doctor of Philosophy
November 2012

Abstract

This thesis is concerned with the numerical approximation of problems of fluid flow, in particular the stationary advection diffusion reaction equations and the time dependent, coupled equations of incompressible miscible displacement in a porous medium.

We begin by introducing the continuous discontinuous Galerkin method for the singularly perturbed advection diffusion reaction problem. This is a method which coincides with the continuous Galerkin method away from internal and boundary layers and with a discontinuous Galerkin method in the vicinity of layers. We prove that this consistent method is stable in the streamline diffusion norm if the convection field flows non-characteristically from the region of the continuous Galerkin to the region of the discontinuous Galerkin method.

We then turn our attention to the equations of incompressible miscible displacement for the concentration, pressure and velocity of one fluid in a porous medium being displaced by another. We show a reliable a posteriori error estimator for the time dependent, coupled equations in the case where the solution has sufficient regularity and the velocity is bounded. We remark that these conditions may not be attained in physically realistic geometries. We therefore present an abstract approach to the stationary problem of miscible displacement and investigate an a posteriori error estimator using weighted spaces that relies on lower regularity requirements for the true solution.

We then return to the continuous discontinuous Galerkin method. We prove in an abstract setting that standard (continuous) Galerkin finite element approximations are the limit of interior penalty discontinuous Galerkin approximations as the penalty parameter tends to infinity. We then show that by varying the penalization parameter on only a subset of the domain we reach the continuous discontinuous method in the limit. We present numerical experiments illustrating this approach both for equations of non-negative characteristic form (closely related to advection diffusion reaction equations) and to the problem of incompressible miscible displacement. We show that we may practically determine appropriate discontinuous and continuous regions, resulting in a significant reduction of the number of degrees of freedom required to approximate a solution, by using the properties of the discontinuous Galerkin approximation to the advection diffusion reaction equation.

We finally present novel code for implementing the continuous discontinuous Galerkin method in C++.

Declaration

The work in this thesis is based on research carried out at the Numerical Analysis Group, the Department of Mathematical Sciences, Durham University, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgements

I would like to thank my supervisor, Dr. Max Jensen. He has provided me with guidance and support throughout my Ph.D. He has offered a disproportionate amount of his time to discussion and explanation, as well as showing superhuman levels of patience when presented with the n^{th} (still incorrect) rewrite of a theorem.

Many thanks also to Dr. James Blowey who as my second supervisor has selflessly agreed to sign forms in Max's absence. The other members of the Numerical Analysis Group have also made me, and the other students, feel very welcome and included in the academic work of the group. Thanks must go particularly to Prof. Brian Straughan who was kind enough to remember my 4^{th} year MMath project and assist me to return to Durham to read for a doctorate.

Outside of Durham special mention goes to Dr. Andrea Cangiani and Dr. Manos Georgoulis at Leicester for many fruitful discussions. Being able to discuss mathematics with other academics has been among the most rewarding aspects of my time in Durham.

I am fortunate to have made many new friends while studying. It is probably true to say that the amount I have learned about numerical analysis is only marginally more than the amount of trivia I have learned from my friends in the coffee room!

Finally all of the thanks remaining will go to my wife, Laura. She unhesitatingly agreed to move from the South to the North East, little understanding the British climate, but never-the-less has remained positive. Without her support I would have been unable to begin, let alone finish.

Contents

Abstract iii

Declaration v

Acknowledgements vi

1 Introduction 1

1.1 The Problems of Interest 1

1.2 Research Objectives 3

1.3 Notation and Useful Lemmas 5

I The cdG Method and its Stability 12

2 Introduction to the Continuous Discontinuous Galerkin Method 13

2.1 The Continuous Galerkin Method and a Motivating Example 13

2.2 The Discontinuous Galerkin Method 17

2.3 The Continuous Discontinuous Galerkin Method 21

3 On the Stability of the cdG Method 26

3.1 Determining the Ω Decomposition 26

3.2 Decoupled and Weighted Formulations 27

3.3 Bounds on the $\tilde{v}_\varepsilon$ Component on $\mathcal{T}_{\mathrm{cG}}$ 32

3.4 Bounds on the $\tilde{v}_0$ Component on $\mathcal{T}_{\mathrm{cG}}$ 34

3.5 An Inf-Sup Condition on $\mathcal{T}_{\mathrm{dG}}$ 35

3.6 Stability of the Decoupled and Weighted Approximations 39

3.7 Numerical Experiments 40

II A Posteriori Error Estimators for IMD 48

4 Introduction to Incompressible Miscible Displacement 49

4.1 Literature Review 49
4.2 The Coefficients of the Problem 51
4.3 Regularity 52
4.4 The RT-dG Finite Element Method 54

5 A Posteriori Error Estimators for RT-dG 57

5.1 Notation and Preliminary Results 57
5.2 An A Posteriori Estimator for the Pressure and Velocity 60
5.3 An A Posteriori Estimator for the Concentration 67
5.4 An A Posteriori Estimator for the Coupled Problem 76

6 A Posteriori Estimators in Weighted Spaces 77

6.1 An Abstract Discussion 77
6.2 Stationary IMD 80
6.3 The Case for an Alternative Approach 89
6.4 Some Results from the Theory of Weighted Spaces 90
6.5 Sobolev Imbeddings in Weighted Spaces 94
6.6 A Posteriori Error Estimators in the Weighted Spaces 96
6.7 A Review of Our Error Estimators 103

III Constraining the Jumps in the dG Method 105

7 On Local Super Penalization 106

7.1 An Abstract Discussion 107
7.2 Non-Negative Characteristic Form 110
7.3 Incompressible Miscible Displacement 114
7.4 Numerical Experiments 119

8 On Determining the $\mathcal{T}_h$ Decomposition **128**

8.1 Continuity and Coercivity 129

8.2 Determining the $\mathcal{T}_h$ Decomposition 133

8.3 Numerical Experiments 138

IV Implementation of cdG **146**

9 Continuous Discontinuous Finite Element Code **147**

9.1 A Note on Implementation in `deal.ii` 147

9.2 Commented Code 151

9.3 Parameter File 194

10 Summary **196**

Glossary of Nomenclature **198**

Bibliography **200**

Chapter 1

Introduction

In this chapter we introduce the problems we will consider throughout this thesis. We also outline our research objectives and describe the structure of the thesis, and then introduce some basic notation and results which we will use frequently.

1.1 The Problems of Interest

We will study two problems: The stationary linear advection diffusion reaction equations; and the non-linear, time dependent, coupled equations of incompressible miscible displacement.

Linear Advection Diffusion Reaction Equations

Consider one fluid flowing through another, such as one chemical being injected into a smooth flowing stream of a second chemical in some industrial process. A simple model of the concentration of the first chemical should describe the random movement (spreading out) of the molecules within the two chemicals (*diffusion*), the bodily movement of the first chemical caused by the flow (*advection* or *convection*) and any interaction between the chemicals removing or adding the species of interest (*reaction*). We do not discuss the derivation of mathematical models for each part but direct interested readers to, e.g., [2]. We consider specifically the case of an incompressible fluid which diffuses isotropically and reaches some steady state. Therefore consider the stationary advection diffusion reaction equation with

homogeneous Dirichlet boundary conditions in d dimensions:

$$-\varepsilon\Delta u + b(\boldsymbol{x})\cdot\nabla u + c(\boldsymbol{x})u = f(\boldsymbol{x}) \qquad \text{for } \boldsymbol{x}\in\Omega\subset\mathbb{R}^d, \tag{1.1.1}$$

$$u = 0 \qquad \text{on } \partial\Omega \tag{1.1.2}$$

with real valued diffusion coefficient $\varepsilon > 0$, advection term $b(\boldsymbol{x}) = \{b_i(\boldsymbol{x})\}$, $i = 1,\ldots,d$ with entries that are Lipschitz continuous real valued functions, real valued reaction term $c(\boldsymbol{x}) \in L^\infty(\Omega)$ and real valued $f \in L^2(\Omega)$. Throughout Δ denotes the Laplacian $\sum_{i=1}^d \partial^2/\partial x_i^2$ and ∇ the divergence operator. This model is also applicable to problems of, e.g., heat transfer and semiconductor physics.

For $0 < \varepsilon \ll 1$ the solution to (1.1.1) typically exhibits boundary or internal layers [62, 106, 107, 117] and we refer to the problem as *singularly perturbed* with perturbation parameter ε. The solution of this type of problem using analytic methods, e.g., [91, 103], has been thoroughly studied but in some cases the techniques employed fail or are inefficient. The study of numerical approaches to these problems is therefore necessary.

Equations of Incompressible Miscible Displacement

We consider the problem of finding the numerical solution to the coupled equations for the pressure $p = p(t,\boldsymbol{x})$, Darcy velocity $u = u(t,\boldsymbol{x})$ and concentration $c = c(t,\boldsymbol{x})$ of one incompressible fluid in a porous medium being displaced by another. We consider the miscible case where both fluids are in the same phase.

Consider the domain $\Omega_T := (0,T]\times\Omega$. The equations for the miscible displacement are given by (e.g., [22, 24])

$$\varphi\frac{\partial c}{\partial t} + u\cdot\nabla c - \nabla\cdot(\mathbb{D}(u)\nabla c) + cq^I = \hat{c}q^I, \tag{1.1.3}$$

$$\nabla\cdot u = q^I - q^P, \tag{1.1.4}$$

$$u = -\frac{\mathbb{K}}{\mu(c)}\left(\nabla p - \rho(c)g\right) \tag{1.1.5}$$

with the boundary conditions on $\partial\Omega_T := (0,T] \times \partial\Omega$ given by

$$u \cdot n = 0, \tag{1.1.6}$$

$$(\mathbb{D}(u)\nabla c) \cdot n = 0 \tag{1.1.7}$$

and the initial conditions

$$c(0,\cdot) = c_0. \tag{1.1.8}$$

We denote by: $\varphi(\boldsymbol{x})$ the porosity of the medium; $q^I \geq 0$ and $q^P \geq 0$ the pressure at injected (source) and production (sink) wells; $\mathbb{K}(\boldsymbol{x})$ the absolute permeability of the medium; $\mu(c)$ the viscosity of the fluid mixture; $\rho(c)$ the density of the fluid mixture; g the constant vector of gravity; $\mathbb{D}(u,\boldsymbol{x})$ the diffusion-dispersion coefficient; $\hat{c}$ the injected concentration; and c_0 the initial concentration. We define $a^{-1}(c) := \mathbb{K}^{-1}\mu$. The coupling is non-linear through the coefficients $\mathbb{D}(u,\boldsymbol{x})$, $\mu(c)$ and the advection term $u \cdot \nabla c$.

This model to describe incompressible miscible displacement has several economically important industrial applications including enhanced oil recovery (EOR) and groundwater flow [25, 94, 97]. In both of these examples an injected fluid (carbon dioxide resp. contaminated water) mixes with a fluid in a reservoir of porous rock filled with a second fluid (oil resp. fresh water). The flow of the injected fluid through the medium is often difficult to measure directly and therefore appropriate numerical models form a major aspect of industrial research in these areas.

1.2 Research Objectives

This thesis has three main objectives:

(O1) To investigate to what extent the additional degrees of freedom in the (interior penalty) discontinuous Galerkin method, compared to the standard continuous Galerkin method, are required for the stability of the finite element approximation to (1.1.1) in the convection dominated regime;

(O2) To present a posteriori error estimators for the finite element approximation to (1.1.3)-(1.1.8) using discontinuous Galerkin methods and to consider in an abstract setting general a posteriori error estimates for the stationary coupled problems, including cases where the domain leads to unbounded solutions;

(O3) To demonstrate that we may practically reduce the number of degrees of freedom required to approximate (1.1.1) compared to the discontinuous Galerkin method, without compromising stability, and to extend these ideas experimentally to (1.1.3)-(1.1.8).

Each of these objectives is fully addressed in this thesis. We split the thesis into four parts, the first three addressing (O1) to (O3) and the fourth detailing the novel code written as part of the numerical experiments. Of course each objective is not distinct and therefore there will be some overlap between parts. A review of relevant previous work in the field will be presented in each part.

In Part I we define and discuss the continuous discontinuous Galerkin (cdG) method and relate it to the continuous Galerkin and discontinuous Galerkin methods. We then proceed to show the stability of the cdG approximation to (1.1.1), given some assumptions, by modifying the bilinear form.

We then in Part II turn our attention to the equations of incompressible miscible displacement. We show an a posteriori error estimator for the coupled approximation on a convex domain where certain regularity requirements are assured. We then present abstract analysis generating a posteriori error estimators for stationary coupled equations. However industrial problems often arise with more complicated domains. After applying the abstract theory to a simple problem with high regularity and using continuous finite elements we discuss the potential of a posteriori error estimators using Babuška Kondratiev spaces.

In Part III we discuss how the cdG method proposed in Part I is related to the discontinuous and continuous methods as the inter-element jumps in the discontinuous method are more heavily penalized. We apply this approach both to the advection diffusion reaction problem and that of incompressible miscible displacement. We also present analysis showing that we can use a discontinuous approximation to

determine an optimum (in a sense to be defined) space in which to find the cdG approximation.

Finally in Part IV we discuss the implementation of the cdG method in C++ and provide annotated code to allow the reconstruction of the various results of this thesis.

1.3 Notation and Useful Lemmas

Here we present our basic notation and recall some standard results. Results are presented without proof, which can be found in most standard finite element texts, e.g., [32, 34, 67]. Specialist notation and results will be introduced in each chapter, and a glossary of selected notation can be found on page 199.

Throughout this thesis C represents a bounded generic constant that may depend on the domain Ω and dimension d, but is independent of other parameters (unless noted). It may change from expression to expression and line to line. The inclusion of an argument, e.g., $C(\gamma)$, shows a factor dependent on the argument, Ω or d, but independent of the other parameters. By $a \lesssim b$ we mean $a \leq Cb$ and similarly for $\gtrsim$.

Sobolev Spaces

We define all integration in the Lebesgue sense and all partial derivatives in the 'weak' sense, i.e., as distributional derivatives. We say a multiindex $\alpha = (\alpha_1, \ldots, \alpha_n)$ where each α_i is a non-negative integer has order $|\alpha| = \alpha_1 + \ldots + \alpha_n$. We will denote by $\partial^j_{x_i}$ the j^{th} partial derivative with respect to coordinate x_i. Then define

$$D^\alpha f(\boldsymbol{x}) := \frac{\partial^{|\alpha|} f(\boldsymbol{x})}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}}.$$

Given Ω a Lebesgue-measurable subset of $\mathbb{R}^d$ with a non-empty interior and a real valued function f on Ω that is Lebesgue measurable we define

$$\|f\|^p_{L^p(\Omega)} := \int_\Omega |f(\boldsymbol{x})|^p \, \mathrm{d}\boldsymbol{x}$$

for $1 \leq p < \infty$ and

$$\|f\|_{L^\infty(\Omega)} := \operatorname{ess\,sup}\{|f(\boldsymbol{x})| : \boldsymbol{x} \in \Omega\}$$

for $p = \infty$. Then the Lebesgue spaces are defined by

$$L^p(\Omega) := \{f : \|f\|_{L^p(\Omega)} < \infty\}. \tag{1.3.1}$$

The set of locally integrable functions is denoted by

$$L^1_{\text{loc}}(\Omega) := \{f : f \in L^1(E), \quad \forall \text{ compact } E \subset \text{ interior } \Omega\}.$$

If $f \in L^1_{\text{loc}}(\Omega)$ and additionally the weak derivatives $D^\alpha f$ exist for all $|\alpha| < m \in \mathbb{Z}$ then we define

$$\|f\|^p_{W^{m,p}(\Omega)} := \sum_{|\alpha|<m} \|D^\alpha f\|^p_{L^p(\Omega)}, \qquad |f|^p_{W^{m,p}(\Omega)} := \sum_{|\alpha|=m} \|D^\alpha f\|^p_{L^p(\Omega)}$$

for $1 \leq p < \infty$ and

$$\|f\|_{W^{m,\infty}(\Omega)} := \max_{|\alpha|<m} \|D^\alpha f\|_{L^\infty(\Omega)}$$

for $p = \infty$. Then the Sobolev spaces are defined by

$$W^{m,p}(\Omega) := \{f \in L^1_{loc}(\Omega) : \|f\|_{W^{m,p}(\Omega)} < \infty\}. \tag{1.3.2}$$

In the case $p = 2$ we use the equivalent notation $H^m(\Omega) \equiv W^{m,2}(\Omega)$.

For vector valued functions $\boldsymbol{\tau} \in [L^p(\Omega)]^d$ we define the norm by

$$\|\boldsymbol{\tau}\|_{[L^p(\Omega)]^d} = \| \, |\boldsymbol{\tau}| \, \|_{L^p(\Omega)}$$

where $|\cdot|$ is the usual vector 2 norm $|\boldsymbol{\tau}| = (\boldsymbol{\tau}\cdot\boldsymbol{\tau})^{1/2}$. To simplify notation we interpret $\|\boldsymbol{\tau}\|_{L^p(\Omega)}$ as $\|\boldsymbol{\tau}\|_{[L^p(\Omega)]^d}$ for vector valued functions. Define

$$H(\text{div};\Omega) = \{v \in [L^2(\Omega)]^d : \nabla \cdot v \in L^2(\Omega)\} \tag{1.3.3}$$

and

$$H_0(\mathrm{div};\Omega) = \{v \in [L^2(\Omega)]^d : \nabla \cdot v \in L^2(\Omega), v \cdot n = 0 \text{ in } H^{-1/2}(\partial\Omega)\}. \tag{1.3.4}$$

The space $L^p((0,T];H^m(\Omega))$ consists of all functions $u : (0,T] \mapsto H^m(\Omega)$ such that $t \mapsto \|u(t)\|_{H^m(\Omega)}$ is in $L^p((0,T])$ with the norm

$$\|u\|^p_{L^p((0,T];H^m(\Omega))} := \int_0^T \|u(t)\|^p_{H^m(\Omega)}\,\mathrm{d}t \tag{1.3.5}$$

for $1 \leq p < \infty$ and

$$\|u\|_{L^\infty((0,T];H^m(\Omega))} := \operatorname*{ess\,sup}_{t\in(0,T]} \|u(t)\|_{H^m(\Omega)} \tag{1.3.6}$$

for $p = \infty$.

The dual of a Banach space V is written V^*.

Triangulations

We assume throughout, unless otherwise stated, that Ω is a bounded, open polygon (polyhedron) in $\mathbb{R}^d$ with a Lipschitz boundary denoted by $\partial\Omega$. For time dependent spaces define $\Omega_T := (0,T] \times \Omega$. Call the subdivision of a domain Ω into non-overlapping shape regular d-simplices E a triangulation $\mathcal{T}_h$, each with boundary denoted ∂E. Denote by $\mathcal{E}_h$ the union of edges e (or faces for $d \geq 3$) of the mesh (the skeleton) and the union of internal edges by $\mathcal{E}_h^o$. Define Γ as the union of elemental boundary edges, i.e., those lying in $\partial\Omega$. The diameter of an element $E \in \mathcal{T}_h$ is denoted h_E and $h = \max_{E\in\mathcal{T}_h} h_E$. Call the diameter of an edge h_e, defined by

$$h_e := \begin{cases} \min(h_{E^+}, h_{E^-}) & \text{for } e = \bar{E}^+ \cap \bar{E}^- \in \mathcal{E}_h^o, \\ h_E & \text{for } \partial E \cap \partial\Omega \in \Gamma \end{cases}$$

for E^+, $E^- \in \mathcal{T}_h$.

We introduce the following notation describing the behaviour of functions that may be discontinuous at interelement boundaries. Given a generic scalar field ν :

$\Omega \to \mathbb{R}$, that may be discontinuous across an edge $e = \bar{E}^+ \cap \bar{E}^-$ for E^+, $E^- \in \mathcal{T}_h$, we set $\nu^\pm := \nu|_{E^\pm}$, the interior trace on $E^\pm$ and similarly define $\boldsymbol{\tau}^\pm = \boldsymbol{\tau}|_{E^\pm}$ for a generic vector field $\boldsymbol{\tau} : \Omega \to \mathbb{R}^d$. Define the average and jump for a generic scalar as

$$\{\!\{\nu\}\!\} := \frac{1}{2}(\nu^+ + \nu^-), \qquad [\![\nu]\!] := \nu^+ n^+ + \nu^- n^-, \qquad \text{on } e \in \mathcal{E}_h^o,$$

and for a generic vector field as

$$\{\!\{\boldsymbol{\tau}\}\!\} := \frac{1}{2}(\boldsymbol{\tau}^+ + \boldsymbol{\tau}^-), \qquad [\![\boldsymbol{\tau}]\!] := \boldsymbol{\tau}^+ \cdot n^+ + \boldsymbol{\tau}^- \cdot n^-, \qquad \text{on } e \in \mathcal{E}_h^o,$$

where $n^\pm$ is the outward pointing normal from $E^\pm$ on e. For $e \in \Gamma$ the definitions become

$$\{\!\{\nu\}\!\} := \nu, \qquad [\![\nu]\!] := \nu n, \qquad \{\!\{\boldsymbol{\tau}\}\!\} := \boldsymbol{\tau}, \qquad [\![\boldsymbol{\tau}]\!] := \boldsymbol{\tau} \cdot n, \qquad \text{on } e \in \Gamma.$$

Note that for an element v from a continuous space we have $[\![v]\!] = 0$ and $\{\!\{v\}\!\} = v$ for every $e \in \mathcal{E}_h^o$.

Given a vector b denote the inflow and outflow boundaries of Ω by

$$\Gamma^{\text{in}} := \{\boldsymbol{x} \in \partial\Omega : b \cdot n \leq 0\},$$
$$\Gamma^{\text{out}} := \{\boldsymbol{x} \in \partial\Omega : b \cdot n > 0\}$$

and for an element

$$\partial^{\text{in}} E := \{\boldsymbol{x} \in \partial E : b \cdot n \leq 0\},$$
$$\partial^{\text{out}} E := \{\boldsymbol{x} \in \partial E : b \cdot n > 0\}.$$

We denote the trace of a function ν on an edge by ν^{in} (resp. ν^{out}) on the side of the edge where $b \cdot n \leq 0$ (resp. $b \cdot n > 0$). We construct the mesh so that the sign of $b \cdot n$ is the same for every $\boldsymbol{x} \in e$.

All meshes are *shape regular*, i.e., there exists $\kappa > 0$ such that every $E \in \mathcal{T}_h$

contains a ball of radius Υ_E with

$$\Upsilon_E \geq \kappa h_E. \tag{1.3.7}$$

This implies that there exists $C > 0$ such that for $e = \partial E^+ \cap \partial E^- \subset \mathcal{E}_h^o$

$$h_e \leq \frac{1}{2}(h_{E^+} + h_{E^-}) \leq C h_e$$

and that there exists a constant $C_{\text{reg}} \geq 1$ such that

$$C_{\text{reg}}^{-1} h_{E^-} \leq h_{E^+} \leq C_{\text{reg}} h_{E^-}. \tag{1.3.8}$$

We will also require in some cases *quasi uniform* meshes. These are meshes where there exists $\kappa > 0$ such that

$$\min_{E \in \mathcal{T}_h} \{\Upsilon_E\} \geq \kappa h.$$

Meshes that are quasi uniform are also shape regular, but the converse does not generally hold.

We denote by $(\,\cdot\,,\,\cdot\,)$ the usual L^2 inner product on Ω, and by $(\,\cdot\,,\,\cdot\,)_E$ and $(\,\cdot\,,\,\cdot\,)_e$ the L^2 inner product on elements and edges respectively.

Frequently Used Lemmas

We include the following inequalities for completeness. All are well known and we state them without proof. We will use them frequently (both with and without reference) throughout this thesis.

Lemma 1.3.9 (Trace inequality)**.** *Suppose that Ω has a Lipschitz boundary and that $1 \leq p \leq \infty$. Then there is a constant C, depending on the shape of Ω and dimension d, such that*

$$\|v\|_{L^p(\partial\Omega)} \leq C \|v\|_{L^p(\Omega)}^{1-1/p} \|v\|_{W^{1,p}(\Omega)}^{1/p} \qquad \forall v \in W^{1,p}(\Omega). \tag{1.3.10}$$

We may also apply this lemma elementwise, in which case C will depend on the mesh regularity, and for $v \in \mathbb{P}^k(E)$, the space of piecewise polynomials on E of degree at most k.

Lemma 1.3.11 (Inverse inequality). *Let $\{\mathcal{T}_h\}$ be a shape regular family of meshes in $\mathbb{R}^d$ with $0 < h \leq 1$. Let V be a finite dimensional subspace of $W^{l,p}(E) \cap W^{m,q}(E)$, where $1 \leq p, q \leq \infty$ and $0 \leq m \leq l$. Then for all $v \in V$ there exists $C > 0$ such that*

$$\|v\|_{W^{l,p}(E)} \leq C h_E^{m-l+d(\frac{1}{p}-\frac{1}{q})} \|v\|_{W^{m,q}(E)}. \tag{1.3.12}$$

where C is dependent on l, m, p, q, the space V, the dimension d and the element E.

Corollary 1.3.13 (Trace inequality for polynomials). *With Lemma 1.3.9 applied elementwise, take $p = 2 = q$ and take k such that $V = \mathbb{P}^k(E) \subseteq H^l(E) \cap H^m(E)$. Then we may use Lemma 1.3.11 to show*

$$\|v\|^2_{L^2(\partial E)} \leq C h_E^{-1} \|v\|^2_{L^2(E)} \quad \forall v \in \mathbb{P}^k(E) \tag{1.3.14}$$

where C depends on the polynomial degree and the mesh regularity.

Lemma 1.3.15 (Hölder's inequality). *For $1 \leq p, q, r \leq \infty$ such that $1 = 1/p + 1/q + 1/r$, if $f \in L^p(\Omega)$, $g \in L^q(\Omega)$ and $h \in L^r(\Omega)$ then $fgh \in L^1(\Omega)$ and*

$$\|fgh\|_{L^1(\Omega)} \leq \|f\|_{L^p(\Omega)} \|g\|_{L^q(\Omega)} \|h\|_{L^r(\Omega)}. \tag{1.3.16}$$

The case $p = q = 2$, $r = \infty$, $h \equiv 1$ is known as the Cauchy-Schwarz inequality and can be written

$$\int_\Omega |fg| \, \mathrm{d}\boldsymbol{x} \leq \|f\|_{L^2(\Omega)} \|g\|_{L^2(\Omega)}. \tag{1.3.17}$$

Lemma 1.3.18 (Minkowski's inequality). *For $1 \leq p \leq \infty$ and $f, g \in L^p(\Omega)$ we have*

$$\|f + g\|_{L^p(\Omega)} \leq \|f\|_{L^p(\Omega)} + \|g\|_{L^p(\Omega)}. \tag{1.3.19}$$

We will regularly apply the following version of Young's inequality.

Lemma 1.3.20 (Young's Inequality)**.** *For any $\delta > 0$ and a, b non-negative real numbers*

$$ab \leq \frac{a^2}{2\delta} + \frac{\delta b^2}{2}. \tag{1.3.21}$$

Lemma 1.3.22 (Poincaré Friedrichs inequality)**.** *Let $1 \leq p \leq \infty$ and let Ω be an open bounded set. Then there exists $C > 0$ such that for all $v \in W^{1,p}(\Omega)$ satisfying additionally $v|_{\partial\Omega} = 0$ in $L^2(\partial\Omega)$ (i.e., the trace of v is 0) we have*

$$\|v\|_{L^p(\Omega)} \leq C\|\nabla v\|_{L^p(\Omega)}. \tag{1.3.23}$$

Finally we present the following relationship which can be found in, e.g., [9].

Lemma 1.3.24. *For a generic scalar $\nu \in L^2(\Omega)$ and vector $\boldsymbol{\tau} \in [L^2(\Omega)]^d$ that may be discontinuous only at interelement boundaries of a triangulation $\mathcal{T}_h$ defined on Ω we may rewrite the sum of the integrals over the mesh skeleton in terms of the jumps and averages as follows:*

$$\sum_{E\in\mathcal{T}_h} \int_{\partial E} \nu\boldsymbol{\tau} \cdot n_E \,\mathrm{d}s = \sum_{e\in\mathcal{E}_h} \int_e [\![\nu]\!] \cdot \{\!\{\boldsymbol{\tau}\}\!\} \,\mathrm{d}s + \sum_{e\in\mathcal{E}_h^o} \int_e \{\!\{\nu\}\!\} [\![\boldsymbol{\tau}]\!] \,\mathrm{d}s. \tag{1.3.25}$$

Part I

The Continuous Discontinuous Galerkin Method and its Stability

Chapter 2

Introduction to the Continuous Discontinuous Galerkin Method

In this chapter we introduce the continuous discontinuous Galerkin (cdG) method for (1.1.1). We first present the continuous Galerkin method and give a simple example to motivate the study of other methods. We then proceed to introduce the interior penalty family of discontinuous Galerkin (dG) methods and review their application to singularly perturbed problems before proposing a method which combines elements of both the cG and dG methods which we call the continuous discontinuous Galerkin (cdG) method.

2.1 The Continuous Galerkin Method and a Motivating Example

The smoothness of the classical solution to (1.1.1)-(1.1.2) depends on the smoothness of the given data see, e.g., [77, 117] for a full discussion. Here we focus on the weak solution of (1.1.1) and its expected regularity for Ω an open, bounded subset of $\mathbb{R}^d$ and the assumptions on the parameters as introduced in Chapter 1.

We multiply (1.1.1) by an element $v \in H_0^1(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}$, where H^m is the usual Sobolev space defined in (1.3.2), and integrate the diffusion term by parts, using the boundary conditions. The bilinear form associated with

(1.1.1) is then given by

$$\mathcal{B}(u,v) := \int_\Omega \varepsilon \nabla u \cdot \nabla v + (b \cdot \nabla u)v + cuv \, \mathrm{d}\boldsymbol{x} \tag{2.1.1}$$

for u, $v \in H_0^1(\Omega)$. We say that $u \in H_0^1(\Omega)$ is a *weak solution* of (1.1.1)-(1.1.2) if

$$\mathcal{B}(u,v) = \int_\Omega fv \, \mathrm{d}\boldsymbol{x}$$

for all $v \in H_0^1(\Omega)$.

We introduce the concept of continuity and coercivity for bilinear forms.

Definition 2.1.2. *A bilinear form $\mathcal{B} : H \times H \to \mathbb{R}$ on a normed linear space H with norm $\|\cdot\|_H$ is* continuous *if there exists a constant Λ_{ct} such that*

$$|\mathcal{B}(v,\hat{v})| \leq \Lambda_{\mathrm{ct}} \|v\|_H \|\hat{v}\|_H \qquad \forall v, \hat{v} \in H \tag{2.1.3}$$

and coercive *on $V \subset H$ if there exists $\Lambda_{\mathrm{cc}} > 0$ such that*

$$\mathcal{B}(v,v) \geq \Lambda_{\mathrm{cc}} \|v\|_H^2 \qquad \forall v \in V. \tag{2.1.4}$$

We may use the Lax-Milgram lemma [67, 98] to show a unique weak solution to the problem provided the bilinear form is elliptic (coercive) and continuous with respect to the norm induced with respect to the usual inner product on $H_0^1(\Omega)$. To ensure the coercivity we make the following standard assumption:

Assumption 2.1.5. *We assume*

$$r(\boldsymbol{x}) := c(\boldsymbol{x}) - \frac{1}{2}\nabla \cdot b(\boldsymbol{x}) \geq \rho > 0 \qquad \forall \boldsymbol{x} \in \Omega \tag{2.1.6}$$

for some $\rho \in \mathbb{R}$.

Higher regularity requires further assumptions on the coefficients and Ω. For a complete description of the regularity of the weak solution and the requirements for additional regularity, see [69, Chapter 6]. Note that a problem with non-zero Dirichlet boundary conditions can easily be transformed into a problem with homogeneous

Dirichlet boundary conditions in the weak setting.

We define the continuous Galerkin (cG) finite element space as follows:

Definition 2.1.7. *Define the continuous Galerkin space to be*

$$V_{cG} := \{v \in H^1(\Omega) : \forall E \in \mathcal{T}_h, v|_E \in \mathbb{P}^k, v|_\Gamma = 0\} \tag{2.1.8}$$

where $\mathbb{P}^k$ is the space of polynomials of degree at most k.

Then the *classical* finite element approximation (or *standard* cG approximation) to (1.1.1) is defined by:

Definition 2.1.9. *Define the continuous Galerkin finite element approximation to (1.1.1)-(1.1.2) as $u_h \in V_{cG}$ satisfying*

$$\begin{aligned} \mathcal{B}_\varepsilon(u_h, v) &:= \sum_{E\in\mathcal{T}_h} \int_E \varepsilon \nabla_h u_h \cdot \nabla_h v + (b \cdot \nabla_h u_h) v + c u_h v \,\mathrm{d}\boldsymbol{x} \\ &= \sum_{E\in\mathcal{T}_h} \int_E f v \,\mathrm{d}\boldsymbol{x} \end{aligned} \tag{2.1.10}$$

for all $v \in V_{cG}$.

Throughout ∇_h refers to the piecewise gradient operator.

The standard cG method is relatively easy to implement. Consider, however, the following one dimensional example.

Example 2.1.1 *Let $\Omega = (0, 1)$. We seek to solve*

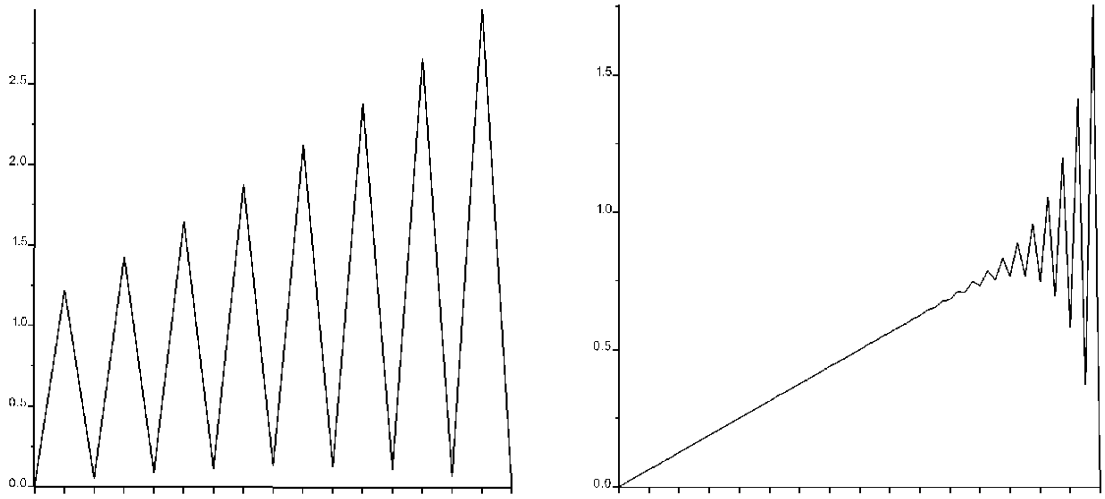
$$-\varepsilon \frac{\mathrm{d}^2}{\mathrm{d}x^2} u(x) + \frac{\mathrm{d}}{\mathrm{d}x} u(x) = 1$$

with boundary conditions $u(0) = u(1) = 0$. The solution is given by

$$u(x) = x - \frac{\mathrm{e}^{(x-1)/\varepsilon} - \mathrm{e}^{-1/\varepsilon}}{1 - \mathrm{e}^{-1/\varepsilon}}.$$

For $0 < \varepsilon \ll 1$ there is a boundary layer at $x = 1$ of width $\mathcal{O}(\varepsilon)$. In Figure 2.1.1 we plot the standard cG approximation to Example 2.1.1 with $\varepsilon = 10^{-3}$ for 16 and 64 uniform intervals (Figures 2.1.1(a) and (b) respectively). In both plots

the oscillations are clear. When we refine the mesh, i.e., use more intervals, the oscillations are less severe (note the scales in each plot) and do not spread throughout the entire region (although they do leave the region of the layer). By refining the interval further we could achieve further reduction in the oscillations, but at the cost of adding degrees of freedom and hence increasing the relative time to generate the approximation.



(a) 16 intervals. (b) 64 intervals.

Figure 2.1.1: Example 2.1.1 with $\varepsilon = 10^{-3}$ on uniformly refined intervals. The oscillations in the second plot are less severe and do not spread throughout the region.

We investigate these oscillations further by considering a simple error estimate for the standard cG method using Céa's lemma [34, (2.8.1)]. First note that $\mathcal{B}_\varepsilon$ has the same continuity and coercivity properties as $\mathcal{B}$ in (2.1.1) for functions in V_{cG}. It is straightforward to show

$$\|u - u_h\|_{H_0^1(\Omega)} \leq \left(\frac{\Lambda_{\text{ct}}}{\Lambda_{\text{cc}}}\right)^{\frac{1}{2}} \inf_{v \in V_{\text{cG}}} \|u - v\|_{H_0^1(\Omega)} \tag{2.1.11}$$

where Λ_{cc} and Λ_{ct} are the coercivity and continuity with respect to the norm $\|\cdot\|_{H_0^1(\Omega)}$ for $\mathcal{B}_\varepsilon$. However the coercivity constant depends adversely on ε and by using a standard interpolation inequality, e.g., [34, Section 4.4] on the right hand side we see that we may bound the error by $\mathcal{O}(h/\varepsilon)|u|_{H^2(\Omega)}$. As $h \to 0$ the error also tends to 0 (for a fixed problem). However as $\varepsilon \to 0$ we do not have a useful bound. In addition we expect $|u|_{H^2(\Omega)}$ to grow as $\varepsilon \to 0$.

2.2 The Discontinuous Galerkin Method

The non-physical oscillations shown for a simple one dimensional example in Figure 2.1.1 are seen also in higher dimensions. Numerical procedures to address this behaviour are too numerous to survey entirely here, and we concentrate on the finite element methods. For alternative approaches, such as finite difference methods, readers are directed to [106, 107, 117] and the references cited therein.

In the field of finite element analysis we crudely divide the attempts to improve the standard cG method into three areas (including combinations of the three): Addition of stabilising terms; adaptation of the mesh; and adaptation of the space. We discuss the first idea only briefly, and the second in more detail in Part II with reference to incompressible miscible displacement. The final idea is of more interest here as it includes the discontinuous and continuous discontinuous Galerkin methods.

In order to stabilize the cG approximation to (1.1.1) we can add weighted *residual* terms to the formulation, that is, terms originating from using a numerical approximation in the original differential equation. In particular the streamline diffusion finite element method, introduced by Hughes and Brooks [84], adds terms generated by applying $b\cdot\nabla_h v$, where $v \in V_{\mathrm{cG}}$ is the test function, to $-\varepsilon\Delta u_h + b\cdot\nabla_h u_h + cu_h - f$. The stability properties are a consequence of the choice of the weight applied, which we will discuss in detail when we apply this approach to the discontinuous Galerkin method in Chapter 8. In short, by adding additional terms to the norm the oscillations can be controlled [118]. See [86, 87] for a thorough study of the choice of weight and further adaptation of the streamline diffusion method. When the true solution u to (1.1.1) satisfies the residual terms exactly we call the resulting numerical method *consistent*. The streamline diffusion method is *non-consistent* as the additional terms mean that the true solution does not satisfy the approximate problem. However when we consider streamline terms in Chapter 3 we only add terms to the norm, not the approximate problem, and so consistency is maintained.

The benefit of adapting the mesh can be seen from the $\mathcal{O}(h/\varepsilon)$ bound in (2.1.11). As $h \to 0$ the error, for a fixed problem, will tend to zero. However such refinement performed globally will be inefficient, adding far more degrees of freedom than re-

quired for stability. More efficient schemes exist for singularly perturbed problems, such as layer adapted meshes, e.g., [106, 122, 129] and anisotropic meshes, e.g., [7, 73, 72, 108]. Schemes for refinement of both mesh size and polynomial degree (so called *hp* refinement) have also been studied extensively in the literature, e.g., [23, 74, 82, 81, 104, 119, 120].

The third approach is of most interest to us in this part. By selecting an approximation from a larger space we hope to be able to eliminate the oscillations associated with the cG approximation. We first focus on the space of piecewise discontinuous polynomials applied on a triangulation $\mathcal{T}_h$. These are polynomials that may be discontinuous at interelement boundaries (including Γ, the exterior boundary).

Definition 2.2.1. *Define the discontinuous Galerkin space to be*

$$V_{dG} := \{v \in L^2(\Omega) : \forall E \in \mathcal{T}_h, v|_E \in \mathbb{P}^k\} \tag{2.2.2}$$

where $\mathbb{P}^k$ is the space of polynomials of degree at most k.

Using discontinuous elements is a form of *variational crime*, so called as the space $V_{\text{dG}} \not\subset H_0^1(\Omega)$. However we do have $V_{\text{cG}} \subset V_{\text{dG}}$, i.e., the dG space is larger than the cG space defined in (2.1.8).

A discussion of discrete formulations for the diffusion term of (1.1.1) for discontinuous spaces can be found in [9]. We consider the interior penalty (IP) family of methods with parameter $\vartheta = \{-1, 1\}$ which are given by, for $w, \hat{w} \in V_{\text{dG}}$,

$$\begin{aligned} &\mathcal{B}_d(w, \hat{w}) := \\ &\sum_{E \in \mathcal{T}_h} \int_E \nabla_h w \cdot \nabla_h \hat{w} \, \mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{e \in \mathcal{E}_h} \int_e \sigma h_e^{-1} [\![w]\!] \cdot [\![\hat{w}]\!] - (\{\!\{\nabla_h w\}\!\} \cdot [\![\hat{w}]\!] - \vartheta \{\!\{\nabla_h \hat{w}\}\!\} \cdot [\![w]\!]) \, \mathrm{d}s \end{aligned} \tag{2.2.3}$$

where $\sigma \in \mathbb{R}$ is the penalty parameter chosen large enough to ensure coercivity of

$\mathcal{B}_d$. We discretize the advection term by

$$\begin{aligned}\mathcal{B}_a(w,\hat{w}) := &\sum_{E\in\mathcal{T}_h}\int_E (b\cdot\nabla_h w)\hat{w}\,\mathrm{d}\boldsymbol{x}\\ &-\sum_{e\in\mathcal{E}_h^o}\int_e b\cdot[\![w]\!]\hat{w}^{\text{out}}\,\mathrm{d}s-\sum_{e\in\Gamma^{\text{in}}}\int_e (b\cdot n)w\hat{w}\,\mathrm{d}s\end{aligned} \tag{2.2.4}$$

and the reaction term by

$$\mathcal{B}_r(w,\hat{w}) := \sum_{E\in\mathcal{T}_h}\int_E cw\hat{w}\,\mathrm{d}\boldsymbol{x}. \tag{2.2.5}$$

The advection and reaction parts will frequently occur together and so for brevity we also define $\mathcal{B}_{ar}(w,\hat{w}) := \mathcal{B}_a(w,\hat{w}) + \mathcal{B}_r(w,\hat{w})$. With these definitions we define the bilinear form $\mathcal{B}_\varepsilon : V_{\text{dG}}\times V_{\text{dG}}\to\mathbb{R}$ by

$$\mathcal{B}_\varepsilon(w,\hat{w}) := \varepsilon\mathcal{B}_d(w,\hat{w}) + \mathcal{B}_a(w,\hat{w}) + \mathcal{B}_r(w,\hat{w}). \tag{2.2.6}$$

Note that we use the same notation $\mathcal{B}_\varepsilon$ for the bilinear form for the dG method as we did in (2.1.10) as (2.2.6) reduces to the standard cG method if we restrict ourselves to elements of V_{cG}.

Definition 2.2.7. *Define the interior penalty discontinuous Galerkin finite element approximation to* (1.1.1)-(1.1.2) *as* $w_h \in V_{dG}$ *satisfying*

$$\mathcal{B}_\varepsilon(w_h,w) = \sum_{E\in\mathcal{T}_h}\int_E fv\,\mathrm{d}\boldsymbol{x} \qquad \forall w\in V_{dG}. \tag{2.2.8}$$

We introduce the mesh dependent norm $|||\cdot|||$ defined by

$$|||w|||^2 := \varepsilon\|w\|_d^2 + \|w\|_{ar}^2, \tag{2.2.9}$$

where

$$\|w\|_d^2 := \sum_{E\in\mathcal{T}_h} |w|_{H^1(E)}^2 + \sum_{e\in\mathcal{E}_h} \sigma h_e^{-1} \|[\![w]\!]\|_{L^2(e)}^2, \tag{2.2.10}$$

$$\|w\|_{ar}^2 := \|r^{1/2}w\|_{L^2(\Omega)}^2 + \sum_{e\in\mathcal{E}_h} \frac{1}{2} \||b\cdot n|^{1/2}[\![w]\!]\|_{L^2(e)}^2. \tag{2.2.11}$$

The discontinuous Galerkin method was first introduced for the first order hyperbolic neutron transport problem by Reed and Hill [114]. For elliptic and parabolic equations the first papers on the interior penalty method include [8, 14, 63, 127], although their development was largely independent of methods for the hyperbolic problem [9]. The idea behind interior penalty methods is that interelement continuity may be weakly enforced by penalizing the jumps (as can be seen in the second term of (2.2.3) with penalty parameter σ). Previous methods had been proposed which weakly enforced Dirichlet boundary conditions [12], and hence the use of the term *interior* penalty. The third term in the interior penalty formulation $\mathcal{B}_d$ arises naturally from integration by parts. The additional term, multiplied by the parameter $\vartheta = \{-1, 1\}$, has a different role depending on the sign. In particular when $\vartheta = -1$ we have the symmetric IP method with the advantage of a symmetric bilinear form (when $b \equiv 0$) but the coercivity constant (with respect to $|||\cdot|||$) depending on the choice of σ. When $\vartheta = 1$ we have the non-symmetric IP method which is unconditionally coercive but lacks adjoint consistency. We do not consider $\vartheta = 0$, the so called incomplete interior penalty method [60].

Several alternative dG methods are common in the literature, such as the local discontinuous Galerkin (LDG) method [55], the method of Bassi and Rebay [21] and that of Baumann and Oden [23]. All of these methods (for the elliptic part) can be brought into the same framework as shown by Arnold et. al. [9]. For a review of the development of the dG methods see [54] and the references included therein. Until recently there were no text books dedicated to discontinuous methods, but increased interest in the topic has led to several publications including [80, 112, 115].

Discontinuous Galerkin methods offer several attractive properties. They admit good stability properties, even for singularly perturbed problems [10, 41] if some streamline terms are added to the norm (but need not be added to the bilinear

form, as we shall see in Chapter 3). They also allow the use of irregular mesh design as hanging nodes are more easily incorporated than for conforming methods. This, along with ready parallelization, allows for relatively simple *hp* adaptivity [82] and a posteriori error estimation [81, 83].

However dG methods also have drawbacks when compared to standard cG methods and their conforming extensions such as the streamline diffusion method. They are (in general) more difficult to implement and require an increased number of degrees of freedom. For instance, when using an axi-parallel quadrilateral mesh in two dimensions with piecewise bilinear elements, for which the standard cG finite element method has approximately n degrees of freedom (depending on the boundary conditions) the dG method on the same mesh has $4n$ degrees of freedom.

2.3 The Continuous Discontinuous Galerkin Method

Given the increased number of degrees of freedom associated with the interior penalty dG method as opposed to the standard cG method we introduce the continuous discontinuous Galerkin (cdG) method. It is our hypothesis that the additional degrees of freedom are only required to achieve stability in regions where the solutions of (1.1.1) have exponential layers. We therefore construct a space which allows discontinuities in the approximation only in the region of layers and enforces continuity elsewhere. To our knowledge this method was first proposed by Cangiani, Georgoulis and Jensen [47] where a comparison was made between the interior penalty dG method, the cdG method, the residual free bubble method [36, 38] and the streamline upwind Petrov-Galerkin (SUPG) method [39, 41, 88].

Conceptually, somewhere between the standard cG and interior penalty dG methods lies the continuous discontinuous Galerkin (cdG) finite element method, whereby one seeks a Galerkin solution on a finite element space V_{cdG} with

$$V_{\text{cG}} \subset V_{\text{cdG}} \subset V_{\text{dG}}. \tag{2.3.1}$$

Behind our work is a question of optimality: Which approximation spaces allow us to formulate a consistent and stable finite element method with a minimal number of degrees of freedom. The motivation of this work is twofold: Firstly we wish to improve our understanding of the current classical dG methods in response to the criticism in the increased number of degrees of freedom. Secondly we seek insight into the design of more efficient finite element methods in the future.

We define by Ω-*decomposition* the splitting of Ω into two regions Ω_{cG} and Ω_{dG} such that for the closure $\overline{\Omega} = \overline{\Omega_{\mathrm{cG}} \cup \Omega_{\mathrm{dG}}}$, and we define by $\mathcal{T}_h$-*decomposition* the splitting of $\mathcal{T}_h$ into two sub-meshes $\mathcal{T}_{\mathrm{cG}}$ and $\mathcal{T}_{\mathrm{dG}}$ such that $\mathcal{T}_{\mathrm{cG}} \subset \Omega_{\mathrm{cG}}$ and $\mathcal{T}_{\mathrm{dG}} := \mathcal{T}_h \setminus \mathcal{T}_{\mathrm{cG}}$. By abuse of language, we denote here by $\mathcal{T}_{\mathrm{cG}}$ not just the sub-mesh but also the region it occupies. When we refer to $\mathcal{T}_{\mathrm{cG}}$ as a region we mean the interior of the closure, i.e.,

$$\mathrm{Int}\left(\overline{\bigcup_{E\in\mathcal{T}_{\mathrm{cG}}} E}\right). \tag{2.3.2}$$

Define Γ_{cG} (resp. Γ_{dG}) to be the intersection of Γ with $\overline{\mathcal{T}}_{\mathrm{cG}}$ (resp. $\overline{\mathcal{T}}_{\mathrm{dG}}$). Define $J := \overline{\mathcal{T}}_{\mathrm{cG}} \cap \overline{\mathcal{T}}_{\mathrm{dG}}$ and by convention we say that the edges lying in J are only part of the discontinuous Galerkin skeleton $\mathcal{E}_{\mathrm{dG}}$, the union of faces in $\overline{\mathcal{T}}_{\mathrm{dG}}$, and not part of the continuous Galerkin skeleton defined by $\mathcal{E}_{\mathrm{cG}} := \mathcal{E}_h \setminus \mathcal{E}_{\mathrm{dG}}$.

In Figure 2.3.1 we illustrate a splitting for a problem where $\Omega = (0,1)^2$ and the solution exhibits layers at $x = 1$ and $y = 1$. The Ω-decomposition is labelled, with the demarcation between the Ω_{cG} and Ω_{dG} regions given by a dashed line. A $\mathcal{T}_h$-decomposition is shown with the $\mathcal{T}_{\mathrm{dG}}$ region shaded and the edges in J marked with a heavy line. According to our hypothesis we place any layers inside the discontinuous region. We discuss the location of Ω_{cG} and Ω_{dG} in Chapter 3 and strategies for determining $\mathcal{T}_{\mathrm{cG}}$ and $\mathcal{T}_{\mathrm{dG}}$ in Chapters 3 and 8.

Definition 2.3.3. *Define the cdG space to be*

$$V_{cdG} := \{v \in L^2(\Omega) : \forall E \in \mathcal{T}_h, v|_E \in \mathbb{P}^k, v|_{\Gamma_{cG}} = 0, v|_{\mathcal{T}_{cG}} \in C(\mathcal{T}_{cG})\}. \tag{2.3.4}$$

Here $C(\mathcal{T}_{\mathrm{cG}})$ is understood in the sense of (2.3.2). Also $v|_{\Gamma_{\mathrm{cG}}} = 0$ means that the

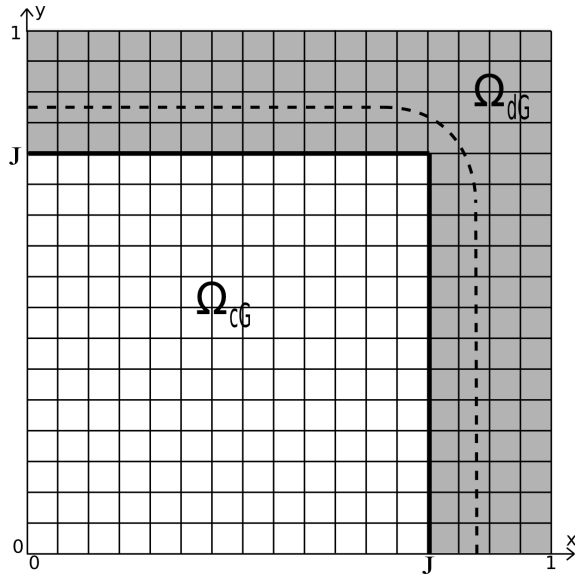


Figure 2.3.1: An example of a cdG decomposition with Ω_{dG} and Ω_{cG} the regions to either side of the dashed line, and $\mathcal{T}_{\mathrm{dG}}$ and $\mathcal{T}_{\mathrm{cG}}$ the shaded and unshaded regions respectively.

trace of v onto Γ_{cG} vanishes. Note that with this definition $V_{\mathrm{cdG}} \subset V_{\mathrm{dG}}$ and $\mathcal{B}_\varepsilon$ defined in (2.2.6) reduces to the form of the standard cG method defined in (2.1.10) on $\mathcal{T}_{\mathrm{cG}}$.

Definition 2.3.5. *Define the interior penalty continuous discontinuous Galerkin finite element approximation to* (1.1.1)-(1.1.2) *as* $v_h \in V_{cdG}$ *satisfying*

$$\mathcal{B}_\varepsilon(v_h, v) = \sum_{E\in\mathcal{T}_h} \int_E f v \,\mathrm{d}\boldsymbol{x} \qquad \forall v \in V_{cdG}. \tag{2.3.6}$$

There is comparatively little published work using this approach. In a broader sense Perugia and Schötzau [111] and Dawson and Proft [59] both consider coupling the local discontinuous Galerkin (LDG) [55] and cG method, in the first case for the Poisson equation, and in the second case for a time dependent transport equation (which may exhibit layers). However in [111] conditions are imposed on the internal boundary J which are then applied for the continuous elements. By imposing weights on the jump at the interface and some further conditions [59] presents a stability result and also error bounds on the cdG approximation. Further examples using the cdG method for the stationary problem can be found in [61], but no additional analysis is presented.

Our approach does not require any transmission conditions between the regions,

other than those which occur naturally through the definition of the jump and the average terms present in $\mathcal{B}_\varepsilon$ defined in (2.2.6).

The paper of Burman and Zunino [43] contains the cdG method proposed here as a special case. The authors propose a scheme with the aim of implementation of the continuous Galerkin scheme for parallel solution using, e.g., multiple processors on a single computer. We will see some aspects of their approach in Chapter 3 where we will weight the averages in the bilinear form $\mathcal{B}_d$ entirely towards the continuous domain, i.e., we redefine $\{\!\{\nabla_h w\}\!\} := \nabla_h w|_{\mathcal{T}_{\mathrm{cG}}}$ for the edges lying in J. The weighting of the averages is also used by Ern, Stephansen and Zunino [68] to consider the case of anisotropic, discontinuous diffusivity by introducing the Symmetric Weighted Interior Penalty (SWIP) method. The authors in both papers remark that by careful selection of the weights the stability properties of the method can be improved, but offer no stability analysis. For convergence analysis and a priori error estimates see in particular [43].

Our construction of V_{cdG} is not the only space that can be chosen to make a cdG type method satisfying $V_{\mathrm{cG}} \subset V_{\mathrm{cdG}} \subset V_{\mathrm{dG}}$. Becker et al. [26] propose a space in which the standard continuous space (2.1.8) is enriched with the space of piecewise constants, i.e., V_{dG} with $k = 0$. Therefore for an axi-parallel quadrilateral mesh in 2 dimensions where the standard cG method has n degrees of freedom using piecewise bilinear finite elements, and the corresponding dG method has $4n$ degrees of freedom, the method of [26] will have $2n$ degrees of freedom. Stability and error estimates are shown. However for problems with sharp layers we will see that we can reduce the number of degrees of freedom to considerably fewer than $2n$ using our proposed method. There may be other choices of V_{cdG} which are more appropriate for other formulations, problems, or with more exotic finite elements.

Additional Notation and Assumptions for cdG

Throughout we assume that we use the same polynomial degree for V_{dG} and V_{cdG}. Let χ be the characteristic function on $\mathcal{T}_{\mathrm{dG}}$, i.e., that defined by

$$\chi := \begin{cases} 1 & \boldsymbol{x} \in \mathcal{T}_{\mathrm{dG}}, \\ 0 & \boldsymbol{x} \in \mathcal{T}_{\mathrm{cG}}. \end{cases} \tag{2.3.7}$$

Then define

$$V_{\mathrm{cdG}}(\mathcal{T}_{\mathrm{dG}}) := \{\chi v : v \in V_{\mathrm{cdG}}\}$$

and

$$V_{\mathrm{cdG}}(\mathcal{T}_{\mathrm{cG}}) := \{(1-\chi)v : v \in V_{\mathrm{cdG}}\}.$$

Definition 2.3.8. *We define the* local mesh Péclet number *[117] to be*

$$P = \frac{\|b\|_{L^\infty(E)} h_E}{2\varepsilon}.$$

We stipulate that the diffusion coefficient satisfies $0 < \varepsilon \leq \varepsilon_{\max}$. We consider meshes in the pre-asymptotic regime. More precisely:

Assumption 2.3.9. *We assume that for $\varepsilon_{\max}$ and every $E \in \mathcal{T}_h$ the local mesh Péclet number is greater than 1. Moreover, we require $\|h_E/b\|_{L^\infty(\mathcal{T}_{dG})} \leq 1$.*

As a consequence we have

$$\varepsilon \leq \varepsilon_{\max} < \frac{1}{2} \min_{E \in \mathcal{T}_h} h_E \|b\|_{L^\infty(\Omega)}. \tag{2.3.10}$$

This assumption, for a fixed b, restricts the refinement of the triangulation for a given ε. If we allowed $h \to 0$ for fixed $\varepsilon > 0$ any layers would be resolved by the mesh and in the limit we would not see the non-physical oscillations associated with the standard cG approximation.

Chapter 3

On the Stability of the Continuous Discontinuous Galerkin Method

In this chapter we discuss the stability of the cdG method. In order to formulate the Ω decomposition of the domain we introduce the *reduced* problem. To show stability we first adapt the bilinear form for the diffusion term (2.2.3) by decoupling and weighting the terms appearing on J, the interface between the continuous and discontinuous regions. We then show the stability of the method on each of $\mathcal{T}_{\text{cG}}$ and $\mathcal{T}_{\text{dG}}$, in the first case by assuming that we have some additional smoothness, and in the second case by proving an inf-sup condition by adapting the approach of [10, 41]. We then combine these results to give a stability result on the whole domain. The material in this chapter is being prepared for publication [44].

3.1 Determining the Ω Decomposition

To characterize admissible Ω-decompositions of the mesh we introduce the *reduced* problem:

$$\begin{aligned} b \cdot \nabla u_0 + c u_0 &= f \quad \text{on } \Omega, \\ u_0 &= 0 \quad \text{on } \Gamma^{\text{in}}. \end{aligned} \tag{3.1.1}$$

This is (1.1.1) with $\varepsilon = 0$ and the boundary conditions adjusted appropriately. We define $u_\varepsilon := u - u_0$, where u is the solution to the ADR equation (1.1.1) and u_0 is

the solution to the reduced problem.

The Ω-decomposition is chosen such that u_ε and u_0 have additional regularity on Ω_{cG}. In general we do not expect that $u_0 \in H^2(\Omega)$, even if we place higher regularity requirements on f, see, e.g., [18] and the references therein.

Assumption 3.1.2. *The set $\Omega_{cG} \subset \Omega$ is chosen such that $u_0 \in H^2(\Omega_{cG})$ and for every $0 < \varepsilon \leq \varepsilon_{\max}$*

$$\|u_\varepsilon\|_{H^2(\Omega_{cG})} \lesssim 1. \tag{3.1.3}$$

The expectation is that the Ω decomposition satisfying Assumption 3.1.2 will consist of a relatively large Ω_{cG}. For the ADR problem with $b = (1, 0)$ Guzmán [78] has shown that we can expect an improved convergence of the dG approximation in L^2 and L^∞, independent of ε, on subdomains $\Omega_0 \subset \Omega$ provided that the boundary of the subdomain is $Ch\log(1/h)$ away from the outflow boundary of Ω. This result points to the additional smoothness of u away from the layers.

3.2 Decoupled and Weighted Formulations

We discretize the advection term by (2.2.4) and the reaction term by (2.2.5). We present two alternative discretizations for the diffusion term, cf., (2.2.3). We first introduce the *decoupled* bilinear form. For $w, \hat{w} \in V_{\mathrm{dG}}$ this is

$$\begin{aligned}&\tilde{\mathcal{B}}_d(w, \hat{w}) := \\ &\sum_{E\in\mathcal{T}_h} \int_E \nabla_h w \cdot \nabla_h \hat{w} \,\mathrm{d}\boldsymbol{x} \\ &\qquad + \sum_{e\in\mathcal{E}_h\setminus J} \int_e \sigma h_e^{-1} [\![w]\!] \cdot [\![\hat{w}]\!] - (\{\!\{\nabla_h w\}\!\} \cdot [\![\hat{w}]\!] + \{\!\{\nabla_h \hat{w}\}\!\} \cdot [\![w]\!]) \,\mathrm{d}s\end{aligned} \tag{3.2.1}$$

and

$$\tilde{\mathcal{B}}_\varepsilon(w, \hat{w}) := \varepsilon\tilde{\mathcal{B}}_d(w, \hat{w}) + \mathcal{B}_a(w, \hat{w}) + \mathcal{B}_r(w, \hat{w}). \tag{3.2.2}$$

With this formulation there is no control on fluxes across J (hence the name decoupled). The utility of this approach will be apparent in Section 3.3. We now introduce the *weighted* bilinear form. On the interface J we make two changes which reduce but do not remove the coupling between the regions. Firstly the average of the trace of the gradients is weighted entirely to the cG side of J (compare this to the approach in [43] where the weighting varies from 0 to 1 on each side of J). With this modification we will not need to consider the upstream gradient on J. Secondly in the penalty term over J the dependence on h is removed. This will allow us at a later point in the analysis to divide by h and still control the jump term. The discretization for the diffusion term is therefore given by

$$\begin{aligned}\overline{\mathcal{B}}_d(w,\hat{w}) &:= \tilde{\mathcal{B}}_d(w,\hat{w}) \\ &\quad + \sum_{e\in J}\int_e \sigma[\![w]\!]\cdot[\![\hat{w}]\!] - (\nabla_h w|_{\mathcal{T}_{\mathrm{cG}}}\cdot[\![\hat{w}]\!] + \nabla_h\hat{w}|_{\mathcal{T}_{\mathrm{cG}}}\cdot[\![w]\!])\,\mathrm{d}s.\end{aligned} \tag{3.2.3}$$

We define the weighted bilinear form by

$$\overline{\mathcal{B}}_\varepsilon(\hat{w},w) := \varepsilon\overline{\mathcal{B}}_d(\hat{w},w) + \mathcal{B}_a(\hat{w},w) + \mathcal{B}_r(\hat{w},w). \tag{3.2.4}$$

When restricted to the cdG space the decoupled and weighted forms become the bilinear form for the standard cG method on the continuous region.

We introduce the following mesh dependent norm for $w \in V_{\mathrm{dG}}$, cf., (2.2.9):

$$|||w|||^2 := \varepsilon\|w\|_d^2 + \|w\|_{ar}^2 \tag{3.2.5}$$

where

$$\|w\|_d^2 := \sum_{E\in\mathcal{T}_h}|w|_{H^1(E)}^2 + \sum_{e\in\mathcal{E}_h\setminus J}\sigma h_e^{-1}\|[\![w]\!]\|_{L^2(e)}^2 + \sum_{e\in J}\sigma\|[\![w]\!]\|_{L^2(e)}^2$$

and $\|w\|_{ar}^2$ is defined in (2.2.11).

Recall that for the symmetric interior penalty method the parameter σ is selected independently of $\mathcal{T}_h$ such that $\mathcal{B}_d$ is positive definite with a coercivity constant which is also independent of $\mathcal{T}_h$.

Assumption 3.2.6. *We assume that σ is such that for all $w \in V_{dG}$ we have*

$$\|\{\!\{\nabla_h w\}\!\} \cdot [\![w]\!]\|_{L^1(\mathcal{E}_h)} \leq \frac{1}{2}\|\nabla_h w\|_{L^2(\Omega)}\|\sqrt{\sigma/h_e}[\![w]\!]\|_{L^2(\mathcal{E}_h)}.$$

Then, by Young's inequality,

$$\frac{1}{2}\|w\|_d^2 \leq \mathcal{B}_d(w,w).$$

We adopt for $\tilde{\mathcal{B}}_d$ and $\overline{\mathcal{B}}_d$ the same σ as for $\mathcal{B}_d$.

We introduce a projection operator following the presentation of [10] which allows us to consider non-constant b. For polynomial degree $k \geq 0$ consider the L^2-orthogonal projection $\Pi_{\mathcal{D}} : L^2(\Omega) \to V_{\mathrm{cdG}}(\mathcal{T}_{\mathrm{dG}})$ defined by

$$\int_\Omega \Pi_{\mathcal{D}}(v) w \,\mathrm{d}\boldsymbol{x} = \int_\Omega vw \,\mathrm{d}\boldsymbol{x} \qquad \forall w \in V_{\mathrm{cdG}}(\mathcal{T}_{\mathrm{dG}}). \tag{3.2.7}$$

In particular $\Pi_{\mathcal{D}}(v)|_{\mathcal{T}_{\mathrm{cG}}} = 0$. Furthermore, the projection has the following property: With $v \in L^2(\Omega)$ and $E \in \mathcal{T}_h$ we have

$$\|\Pi_{\mathcal{D}}(v)\|_{L^2(E)} \leq C\|v\|_{L^2(E)} \qquad \forall v \in L^2(E) \tag{3.2.8}$$

where C is independent of h but depends on the approximation properties of the projection. As $\Pi_{\mathcal{D}}(v) \in V_{\mathrm{cdG}}(\mathcal{T}_{\mathrm{dG}})$ we have for all $E \in \mathcal{T}_h$ the inverse inequality

$$|\Pi_{\mathcal{D}}(b \cdot \nabla_h v)|_{H^1(E)} \lesssim h_E^{-1}\|\Pi_{\mathcal{D}}(b \cdot \nabla_h v)\|_{L^2(E)} \tag{3.2.9}$$

and using a trace inequality we have

$$\sum_{e \in \mathcal{E}_h} \|[\![\Pi_{\mathcal{D}}(b \cdot \nabla_h v)]\!]\|_{L^2(e)}^2 \lesssim \sum_{E \in \mathcal{T}_h} h_E^{-1}\|\Pi_{\mathcal{D}}(b \cdot \nabla_h v)\|_{L^2(E)}^2. \tag{3.2.10}$$

Define the *streamline* norm by

$$\|v\|_S^2 := |||v|||^2 + \sum_{E \in \mathcal{T}_h} \tau_E \|\Pi_{\mathcal{D}}(b \cdot \nabla_h v)\|_{L^2(E)}^2 \tag{3.2.11}$$

where τ_E is defined by

$$\tau_E := \tau \min\left\{ \frac{h_E}{\|b\|_{L^\infty(E)}}, \frac{h_E^2}{\varepsilon} \right\} \tag{3.2.12}$$

and τ is a positive number at our disposal.

Definition 3.2.13. *A decoupled cdG approximation to (1.1.1) is defined as $\tilde{v}_h \in V_{cdG}$ satisfying*

$$\tilde{\mathcal{B}}_\varepsilon(\tilde{v}_h, v) = \int_\Omega f v \, \mathrm{d}\boldsymbol{x} \qquad \forall v \in V_{cdG}. \tag{3.2.14}$$

Definition 3.2.15. *A weighted cdG approximation to (1.1.1) is defined as $\overline{v}_h \in V_{cdG}$ satisfying*

$$\overline{\mathcal{B}}_\varepsilon(\overline{v}_h, v) = \int_\Omega f v \, \mathrm{d}\boldsymbol{x} \qquad \forall v \in V_{cdG}. \tag{3.2.16}$$

We require that b points on J non-characteristically from $\mathcal{T}_{\mathrm{cG}}$ to $\mathcal{T}_{\mathrm{dG}}$.

Assumption 3.2.17. *The $\mathcal{T}_h$ decomposition is such that for every $e \in J$*

$$\frac{1}{4}(b(\boldsymbol{x}) \cdot n^{\mathcal{C}})|_e > \varepsilon_{\max} \frac{\sigma}{h_e} \qquad \forall \boldsymbol{x} \in e \tag{3.2.18}$$

where $n^{\mathcal{C}}$ represents the unit normal pointing from $\mathcal{T}_{cG}$ to $\mathcal{T}_{dG}$.

Lemma 3.2.19. *On V_{dG} the bilinear forms $\tilde{\mathcal{B}}_\varepsilon$ and $\overline{\mathcal{B}}_\varepsilon$ are coercive with respect to $|||w|||$:*

$$\frac{1}{4}|||w|||^2 \le \tilde{\mathcal{B}}_\varepsilon(w, w), \qquad \frac{1}{4}|||w|||^2 \le \overline{\mathcal{B}}_\varepsilon(w, w), \qquad w \in V_{dG}. \tag{3.2.20}$$

Proof. For the advection and reaction terms using integration by parts we have

$$\begin{aligned}\mathcal{B}_{ar}(w,w) = &\sum_{E\in\mathcal{T}_h}\int_E -\frac{1}{2}(\nabla_h\cdot b)w^2\,\mathrm{d}\boldsymbol{x} + \int_{\partial E}\frac{1}{2}(b\cdot n)w^2\,\mathrm{d}s\\ &-\sum_{e\in\mathcal{E}_h^o}\int_e b\cdot[\![w]\!]w^{\mathrm{out}}\,\mathrm{d}s - \sum_{e\in\Gamma^{\mathrm{in}}}\int_e (b\cdot n)w^2\,\mathrm{d}s\\ &+\sum_{E\in\mathcal{T}_h}\int_E cw^2\,\mathrm{d}\boldsymbol{x}\\ =&\sum_{E\in\mathcal{T}_h}\int_E (c-\frac{1}{2}\nabla_h\cdot b)w^2\,\mathrm{d}\boldsymbol{x} + \sum_{e\in\mathcal{E}_h}\int_e\frac{1}{2}(b\cdot n)[\![w]\!]\cdot[\![w]\!]\,\mathrm{d}s.\end{aligned} \tag{3.2.21}$$

For the diffusion term it follows from Assumption 3.2.6 and Young's inequality that

$$\tilde{\mathcal{B}}_d(w,w)+\int_J\frac{\sigma}{h_e}[\![w]\!]\cdot[\![w]\!]\,\mathrm{d}s\geq\frac{1}{2}\|w\|_d^2,\qquad \overline{\mathcal{B}}_d(w,w)+\int_J\frac{\sigma}{h_e}[\![w]\!]\cdot[\![w]\!]\,\mathrm{d}s\geq\frac{1}{2}\|w\|_d^2.$$

Combining the last inequality with (3.2.21), the result now follows with Assumption 3.2.17. □

It follows that $\tilde{v}_h$ and $\overline{v}_h$ exist and are unique. The final assumption permits the use of an inverse inequality on the continuous Galerkin region. It is convenient to define $h_{\mathcal{T}_{\mathrm{cG}}} := \|h_E\|_{L^\infty(\mathcal{T}_{\mathrm{cG}})}$.

Assumption 3.2.22. *The mesh $\mathcal{T}_{cG}$ is quasi-uniform.*

We define $\tilde{v}_\varepsilon, \tilde{v}_0 \in V_{\mathrm{cdG}}$ by the condition that for all $v \in V_{\mathrm{cdG}}$

$$\tilde{\mathcal{B}}_\varepsilon(\tilde{v}_\varepsilon, v) = \tilde{\mathcal{B}}_\varepsilon(u_\varepsilon, v), \tag{3.2.23}$$

$$\tilde{\mathcal{B}}_\varepsilon(\tilde{v}_0, v) = \tilde{\mathcal{B}}_\varepsilon(u_0, v). \tag{3.2.24}$$

Observe that by linearity of the decoupled cdG method we have $\tilde{v}_\varepsilon + \tilde{v}_0 = \tilde{v}_h$. We now proceed to bound each of $\tilde{v}_\varepsilon$ and $\tilde{v}_0$ on $\mathcal{T}_{\mathrm{cG}}$.

3.3 Bounds on the $\tilde{v}_\varepsilon$ Component on $\mathcal{T}_{\mathbf{cG}}$

We introduce the projection operator of Scott and Zhang, e.g., [121] and [67, Section 1.6.2].

Lemma 3.3.1. *The Scott-Zhang operator $\mathcal{SZ}_h : W^{l,p}(\Omega) \to V_{cG}$ is a mapping with the following properties: For $l > \frac{1}{2}$ there exists a $C_{\mathrm{sz}} > 0$ such that for all $0 \le m \le \min(1, l)$*

$$\|\mathcal{SZ}_h(v)\|_{H^m(\mathcal{T}_{cG})} \le C_{\mathrm{sz}} \|v\|_{H^l(\mathcal{T}_{cG})} \qquad \forall v \in H^l(\mathcal{T}_{cG}) \tag{3.3.2}$$

and provided $l \le k+1$ for all $E \in \mathcal{T}_{cG}$ and $0 \le m \le l$ we have the approximation

$$\|v - \mathcal{SZ}_h(v)\|_{H^m(E)} \le C_{\mathrm{sz}} h_E^{l-m} |v|_{H^l(\Delta_E)} \qquad \forall v \in H^l(\Delta_E). \tag{3.3.3}$$

where Δ_E is the node patch of E, i.e., the set of cells in $\mathcal{T}_{cG}$ sharing at least one vertex with E.

Theorem 3.3.4. *The decoupled cdG approximation $\tilde{v}_\varepsilon$ is stable on the $\mathcal{T}_{cG}$ region in the sense that*

$$\|\tilde{v}_\varepsilon\|_{H^1(\mathcal{T}_{cG})} \lesssim 1. \tag{3.3.5}$$

Proof. We pick the *auxiliary* solution $v_{\mathcal{A}}$ to be the Scott-Zhang projection of u_ε on $\mathcal{T}_{\mathrm{cG}}$ and on the restriction to $\mathcal{T}_{\mathrm{dG}}$ to be the dG approximation with boundary conditions given by $\mathcal{SZ}_h(u_\varepsilon)$ on $\Gamma_{\mathrm{dG}} \cup J$, i.e.,

$$\begin{aligned} v_{\mathcal{A}} &= \mathcal{SZ}_h(u_\varepsilon) \quad \text{on } \mathcal{T}_{\mathrm{cG}}, \\ \tilde{\mathcal{B}}_\varepsilon(v_{\mathcal{A}}, v) &= \tilde{\mathcal{B}}_\varepsilon(u_\varepsilon, v) \quad \forall v \in V_{\mathrm{cdG}}(\mathcal{T}_{\mathrm{dG}}). \end{aligned}$$

Set $\eta := u_\varepsilon - v_{\mathcal{A}}$ and $\xi := v_{\mathcal{A}} - \tilde{v}_\varepsilon$, so $\eta + \xi = u_\varepsilon - \tilde{v}_\varepsilon$. Notice that $\xi \in V_{\mathrm{cdG}}$. The Galerkin orthogonality of (3.2.23) and Lemma 3.2.19 give

$$\frac{1}{4}|||\xi|||^2 \le \tilde{\mathcal{B}}_\varepsilon(\xi, \xi) = -\tilde{\mathcal{B}}_\varepsilon(\eta, \xi) = -\tilde{\mathcal{B}}_\varepsilon(\eta, \xi - \chi\xi) \tag{3.3.6}$$

where χ is the characteristic function defined in (2.3.7). Note that $\xi - \chi\xi$ is continuous except on J where $[\![\xi - \chi\xi]\!] = \xi^{\mathcal{C}} \cdot n^{\mathcal{C}}$ and $\{\!\{\xi - \chi\xi\}\!\} = {}^1\!/_2\xi^{\mathcal{C}}$, where the superscript $\mathcal{C}$ indicates the trace taken from the continuous Galerkin side of J.

We examine each term of $\tilde{\mathcal{B}}_\varepsilon$ in turn. For the diffusion parts we use Young's inequality

$$-\tilde{\mathcal{B}}_d(\eta, \xi - \chi\xi) \leq 2|\eta|^2_{H^1(\mathcal{T}_{\mathrm{cG}})} + \frac{1}{8}|\xi|^2_{H^1(\mathcal{T}_{\mathrm{cG}})}.$$

For the advection term we use Assumption 3.2.17 which ensures that flux terms on J are zero as the upwind value of $\xi - \chi\xi$ vanishes. With Young's inequality we have

$$-\mathcal{B}_a(\eta, \xi - \chi\xi) \leq \frac{4}{\rho}\|b \cdot \nabla_h \eta\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{\rho}{16}\|\xi\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})}$$

where ρ is defined in (2.1.6). Finally for the reaction term

$$-\mathcal{B}_r(\eta, \xi - \chi\xi) \leq \frac{4}{\rho}\|c\|^2_{L^\infty(\Omega)}\|\eta\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{\rho}{16}\|\xi\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})}.$$

Using the previous three results, (3.3.6), the definition of the norm (3.2.5), and Lemma 3.3.1 we gather ξ terms on the left hand side to show, recalling that $h_{\mathcal{T}_{\mathrm{cG}}} = \|h_E\|_{L^\infty(\mathcal{T}_{\mathrm{cG}})}$,

$$\begin{aligned}\frac{1}{8}|||\xi|||^2 &\leq 2\varepsilon|\eta|^2_{H^1(\mathcal{T}_{\mathrm{cG}})} + \frac{4}{\rho}\|b \cdot \nabla_h \eta\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{4}{\rho}\|c\|^2_{L^\infty(\Omega)}\|\eta\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} \\ &\lesssim (\varepsilon h^2_{\mathcal{T}_{\mathrm{cG}}} + h^2_{\mathcal{T}_{\mathrm{cG}}} + h^4_{\mathcal{T}_{\mathrm{cG}}})\|u_\varepsilon\|^2_{H^2(\Omega_{\mathrm{cG}})} \lesssim h^2_{\mathcal{T}_{\mathrm{cG}}} \qquad (3.3.7)\end{aligned}$$

where in the final step we have used (3.1.3). As $\rho > 0$ we may use (3.3.7) and an inverse inequality to show

$$\|\xi\|^2_{H^1(\mathcal{T}_{\mathrm{cG}})} \lesssim h^{-2}_{\mathcal{T}_{\mathrm{cG}}}\|\xi\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} \lesssim h^{-2}_{\mathcal{T}_{\mathrm{cG}}}|||\xi|||^2 \lesssim 1. \qquad (3.3.8)$$

Assumption 3.1.2 and (3.3.2) give $\|\tilde{v}_\varepsilon\|^2_{H^1(\mathcal{T}_{\mathrm{cG}})} \lesssim 1$. □

3.4 Bounds on the $\tilde{v}_0$ Component on $\mathcal{T}_{\mathrm{cG}}$

We now pick the auxiliary solution $v_{\mathcal{A}}$ to be u_0 on $\mathcal{T}_{\mathrm{cG}}$ and on $\mathcal{T}_{\mathrm{dG}}$ to be the dG approximation to u_0 with boundary conditions given by u_0 on $\Gamma_{\mathrm{dG}} \cup J$, i.e.,

$$v_{\mathcal{A}} = u_0 \qquad \text{on } \mathcal{T}_{\mathrm{cG}}, \tag{3.4.1}$$

$$\tilde{\mathcal{B}}_\varepsilon(v_{\mathcal{A}}, v) = \tilde{\mathcal{B}}_\varepsilon(u_0, v) \qquad \forall v \in V_{\mathrm{cdG}}(\mathcal{T}_{\mathrm{dG}}). \tag{3.4.2}$$

Lemma 3.4.3. *We have for all $v \in V_{cdG}$ that $\tilde{\mathcal{B}}_\varepsilon(v_{\mathcal{A}}, v) = \tilde{\mathcal{B}}_\varepsilon(\tilde{v}_0, v)$.*

Proof. Fix $v \in V_{\mathrm{cdG}}$. Then using (3.2.24)

$$\tilde{\mathcal{B}}_\varepsilon(\tilde{v}_0, v) = \tilde{\mathcal{B}}_\varepsilon(u_0, v) = \tilde{\mathcal{B}}_\varepsilon(u_0, v - \chi v) + \tilde{\mathcal{B}}_\varepsilon(u_0, \chi v)$$

where χ is defined in (2.3.7). Observe that $\tilde{\mathcal{B}}_\varepsilon(u_0, \chi v) = \tilde{\mathcal{B}}_\varepsilon(v_{\mathcal{A}}, \chi v)$ by (3.4.2). Notice that $v - \chi v$ and u_0 are continuous on $\mathcal{T}_{\mathrm{cG}}$. Recall that integrands over J do not appear in the definition of $\tilde{\mathcal{B}}_d$. For $\tilde{\mathcal{B}}_a(\tilde{v}_0, v)$, the integral over J vanishes since the value of $(v - \chi v)^{\mathrm{out}}$ is zero because of Assumption 3.2.17. Therefore $\tilde{\mathcal{B}}_\varepsilon(\tilde{v}_0, v - \chi v) = \tilde{\mathcal{B}}_\varepsilon(u_0, v - \chi v) = \tilde{\mathcal{B}}_\varepsilon(v_{\mathcal{A}}, v - \chi v)$. □

Lemma 3.4.4. *We have $\|\tilde{v}_0\|_{H^1(\mathcal{T}_{cG})} \lesssim 1$.*

Proof. Define $\tilde{v}_\pi$ to be

$$\tilde{v}_\pi := \begin{cases} \mathcal{SZ}_h(u_0) & \text{on } \mathcal{T}_{\mathrm{cG}}, \\ v_{\mathcal{A}} & \text{on } \mathcal{T}_{\mathrm{dG}} \end{cases}$$

and let $\eta := v_{\mathcal{A}} - \tilde{v}_\pi$, $\xi := \tilde{v}_\pi - \tilde{v}_0$. With these definitions $\eta + \xi = v_{\mathcal{A}} - \tilde{v}_0$, $\eta|_{\mathcal{T}_{\mathrm{dG}}} = 0$ and ξ and η are continuous on $\mathcal{T}_{\mathrm{cG}}$. Then using Lemma 3.4.3 we have

$$\begin{aligned} \frac{1}{4}|||\xi|||^2 \le \tilde{\mathcal{B}}_\varepsilon(\xi, \xi) &= -\tilde{\mathcal{B}}_\varepsilon(\eta, \xi) \\ &= -\int_{\mathcal{T}_{\mathrm{cG}}} \varepsilon \nabla_h \eta \cdot \nabla_h \xi + (b \cdot \nabla_h \eta)\xi + c\eta\xi \, \mathrm{d}\boldsymbol{x} + \int_J b \cdot [\![\eta]\!] \xi^{\mathrm{out}} \, \mathrm{d}s. \end{aligned}$$

Due to Assumption 3.2.17 we have $\xi^{\mathrm{out}} = \xi^{\mathcal{D}}$, the trace from the dG side of J, and $[\![\eta]\!] = \eta^{\mathcal{C}} n^{\mathcal{C}}$, the trace and normal from the cG side of J. We split each of the terms

using Young's inequality, giving

$$\frac{1}{4}|||\xi|||^2 \leq 2\varepsilon\|\nabla_h\eta\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{\varepsilon}{8}\|\nabla_h\xi\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{4}{\rho}\|b\cdot\nabla_h\eta\|_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{\rho}{16}\|\xi\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{4}{\rho}\|c\|^2_{L^\infty(\Omega)}\|\eta\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{\rho}{16}\|\xi\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \int_J (b\cdot n^{\mathcal{C}}\eta^{\mathcal{C}})\xi^{\mathcal{D}}\,\mathrm{d}s.$$

For the final term we note that ξ is a polynomial and so using Young's inequality and a trace and inverse inequality (with constant C_{ti}) gives

$$\int_J (b\cdot n^{\mathcal{C}}\eta^{\mathcal{C}})\xi^{\mathcal{D}}\,\mathrm{d}s \leq \frac{4C_{\mathrm{ti}}\|b\|^2_{L^\infty(\Omega)}}{h_e\rho}\|\eta^{\mathcal{C}}\|^2_{L^2(J)} + \frac{\rho}{16}\|\xi\|^2_{L^2(\mathcal{T}_{\mathrm{dG}})}.$$

Using (3.3.3) and a trace inequality gives

$$\rho\|\xi\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} \leq |||\xi|||^2 \lesssim (\varepsilon h^2_{\mathcal{T}_{\mathrm{cG}}} + h^4_{\mathcal{T}_{\mathrm{cG}}} + h^2_{\mathcal{T}_{\mathrm{cG}}})\|u_0\|^2_{H^2(\mathcal{T}_{\mathrm{cG}})} \lesssim h^2_{\mathcal{T}_{\mathrm{cG}}}\|u_0\|^2_{H^2(\mathcal{T}_{\mathrm{cG}})}$$

and, by an inverse inequality, $\|\xi\|^2_{H^1(\mathcal{T}_{\mathrm{cG}})} \lesssim 1$. Now the result follows from the stability of the Scott-Zhang operator. □

3.5 An Inf-Sup Condition on $\mathcal{T}_{\mathrm{dG}}$

The following theorem is an adaptation of related stability bounds in [41] and [10] to fit the above assumptions. Indeed while the proof of the below inf-sup condition follows the overall structure in [41] closely, we state it here in detail: It extends the scope to non-constant advection coefficients via the incorporation of $\Pi_{\mathcal{D}}$ as [10]; it deals with the modification of the bilinear form and streamline norm on J and it only has streamline control on the $\mathcal{T}_{\mathrm{dG}}$ side. It is helpful to recall that $\Pi_{\mathcal{D}}v|_{\mathcal{T}_{\mathrm{cG}}} = 0$ for any v.

Theorem 3.5.1. *There exists a positive constant Λ_{is} which is independent of h, and ε but may depend on the polynomial degree, σ and the constants in (3.2.9) and (3.2.10) such that:*

$$\inf_{v\in V_{cdG}} \sup_{\hat{v}\in V_{cdG}} \frac{\tilde{\mathcal{B}}_\varepsilon(v,\hat{v})}{\|v\|_S\|\hat{v}\|_S} \geq \Lambda_{\mathrm{is}}. \tag{3.5.2}$$

Proof. Pick an arbitrary $v \in V_{\mathrm{cdG}}$. Then define

$$\hat{v} := v + \gamma v_S, \qquad v_S := \sum_{E\in\mathcal{T}_h} \tau_E \Pi_{\mathcal{D}}(b\cdot\nabla_h v) \tag{3.5.3}$$

where γ is a positive parameter at our disposal and τ_E is defined in (3.2.12). Note that through the definition of $\Pi_{\mathcal{D}}$ we have $\hat{v}, v_S \in V_{\mathrm{cdG}}$. Theorem 3.5.1 is equivalent to showing the following two results:

$$\|\hat{v}\|_S \lesssim \|v\|_S, \tag{3.5.4}$$

$$\tilde{\mathcal{B}}_\varepsilon(v,\hat{v}) \gtrsim \|v\|_S^2. \tag{3.5.5}$$

Consider first (3.5.4). We examine each term of $\|v_S\|_S^2$ in turn. We have

$$\begin{aligned}\sum_{E\in\mathcal{T}_h} \varepsilon |v_S|_{H^1(E)}^2 &\lesssim \sum_{E\in\mathcal{T}_h} \varepsilon h_E^{-2} \|\tau_E \Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|_{L^2(E)}^2 \\ &\le \sum_{E\in\mathcal{T}_h} \tau\tau_E \|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|_{L^2(E)}^2 \lesssim \|v\|_S^2.\end{aligned} \tag{3.5.6}$$

Also

$$\|r^{1/2} v_S\|_{L^2(\Omega)}^2 \le \|r\|_{L^\infty(\Omega)} \sum_{E\in\mathcal{T}_h} \tau_E^2 \|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|_{L^2(E)}^2 \lesssim \|v\|_S^2. \tag{3.5.7}$$

For the terms on the edges we use (3.2.10). This gives

$$\sum_{e\in\mathcal{E}_h} \||b\cdot n|^{1/2} [\![v_S]\!]\|_{L^2(e)}^2 \lesssim \sum_{E\in\mathcal{T}_h} \|b\|_{L^\infty(\Omega)} \tau_E^2 h_E^{-1} \|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|_{L^2(E)}^2 \lesssim \|v\|_S^2. \tag{3.5.8}$$

Similarly,

$$\begin{aligned}&\sum_{e\in J} \sigma\varepsilon \|[\![v_S]\!]\|_{L^2(e)}^2 + \sum_{e\in\mathcal{E}_h\setminus J} \frac{\sigma\varepsilon}{h_e} \|[\![v_S]\!]\|_{L^2(e)}^2 \\ &\lesssim \sum_{E\in\mathcal{T}_h} \tau_E^2 \frac{\sigma\varepsilon}{h_E^2} \|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|_{L^2(E)}^2 \lesssim \|v\|_S^2.\end{aligned} \tag{3.5.9}$$

The final term of the streamline norm gives

$$\sum_{E\in\mathcal{T}_h}\tau_E\|\Pi_{\mathcal{D}}(b\cdot\nabla_h v_S)\|^2_{L^2(E)} \leq \sum_{E\in\mathcal{T}_h}\tau_E\|b\cdot\nabla_h\left(\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\right)\|^2_{L^2(E)}$$
$$\lesssim \sum_{E\in\mathcal{T}_h}\tau_E^3\|b\|^2_{L^\infty(E)}h_E^{-2}\|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|^2_{L^2(E)} \lesssim \|v\|_S^2.$$

Combining the above results we have $\|v_S\|_S^2 \lesssim \|v\|_S^2$. Using a triangle inequality we find

$$\|\hat{v}\|_S \leq \|v\|_S + \gamma\|v_S\|_S \leq C(\tau,\sigma,\gamma)\|v\|_S,$$

which concludes the proof of (3.5.4).

To prove (3.5.5) first consider the advection and reaction terms of the norm. Using the linearity of $\mathcal{B}_{ar}$ we have $\mathcal{B}_{ar}(v,\hat{v}) = \mathcal{B}_{ar}(v,v) + \gamma\mathcal{B}_{ar}(v,v_S)$. The second term equals

$$\mathcal{B}_{ar}(v,v_S) = \sum_{E\in\mathcal{T}_h}\int_E cv(\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v)) + (b\cdot\nabla_h v)(\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v))\,\mathrm{d}\boldsymbol{x}$$
$$-\sum_{e\in\mathcal{E}_h^o}\int_e b\cdot[\![v]\!](\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v))^{\text{out}}\,\mathrm{d}s$$
$$-\sum_{e\in\Gamma^{\text{in}}}\int_e (b\cdot n)v(\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v))\,\mathrm{d}s.$$

Using the properties of $\Pi_{\mathcal{D}}$ given in (3.2.7) the second term above becomes

$$\begin{aligned}
&\sum_{E\in\mathcal{T}_h}\int_E (b\cdot\nabla_h v)(\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v))\,\mathrm{d}\boldsymbol{x}\\
&= \sum_{E\in\mathcal{T}_h}\int_E \tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\,\mathrm{d}\boldsymbol{x}\\
&= \sum_{E\in\mathcal{T}_h}\tau_E\|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|^2_{L^2(E)}.
\end{aligned} \tag{3.5.10}$$

Using Young's inequality we have

$$\Big|\sum_{E\in\mathcal{T}_h}\int_E cv(\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v))\,\mathrm{d}\boldsymbol{x}\Big| \leq \sum_{E\in\mathcal{T}_h}\frac{1}{2}\|c\|_{L^\infty(E)}\|v\|^2_{L^2(E)} + \frac{1}{2}\tau_E^2\|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|^2_{L^2(E)}$$

and, where C arises from a trace inequality and the number of edges per element,

$$-\sum_{e\in\mathcal{E}_h^o}\int_e b\cdot[\![v]\!](\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v))^{\mathrm{out}}\,\mathrm{d}s-\sum_{e\in\Gamma^{\mathrm{in}}}\int_e (b\cdot n)v(\tau_E\Pi_{\mathcal{D}}(b\cdot\nabla_h v))\,\mathrm{d}s$$
$$\leq\sum_{e\in\mathcal{E}_h}\frac{C\lambda}{2}\||b\cdot n|^{1/2}[\![v]\!]\|^2_{L^2(e)}+\sum_{E\in\mathcal{T}_h}\frac{\tau_E\tau}{2\lambda}\|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|^2_{L^2(E)}.$$

In conclusion, together with (3.2.21),

(3.5.11)
$$\mathcal{B}_{ar}(v,\hat{v})\geq\left(\rho-\frac{\gamma\|c\|_{L^\infty(\Omega)}}{2}\right)\sum_{E\in\mathcal{T}_h}\|v\|^2_{L^2(E)}+\left(\frac{1}{2}-\frac{\gamma C\lambda}{2}\right)\sum_{e\in\mathcal{E}_h}\||b\cdot n|^{1/2}[\![v]\!]\|^2_{L^2(e)}$$
$$+\gamma\sum_{E\in\mathcal{T}_h}\left(\tau_E-\frac{\tau_E^2}{2}-\frac{\tau_E\tau}{2\lambda}\right)\|\Pi_{\mathcal{D}}(b\cdot\nabla_h v)\|^2_{L^2(\Omega)}.$$

Recall that $\|h_E/b\|_{L^\infty(\mathcal{T}_{\mathrm{dG}})}\leq 1$ via Assumption 2.3.9. Therefore $\tau_E\leq 1$ for all $E\in\mathcal{T}_h$ by (3.2.12). For general v, all terms on the right-hand side of (3.5.11) are positive, provided λ is large enough, γ is small enough and $\gamma\lambda$ is small enough.

Assumption 3.2.6 ensures the continuity of $\tilde{\mathcal{B}}_d$ with respect to $\|\cdot\|_d$; thus

(3.5.12)
$$\tilde{\mathcal{B}}_d(v,\hat{v})\leq C_1\|v\|_d\|\hat{v}\|_d$$

for some $C_1>0$. Note that the weaker penalisation of jumps on J does not cause a problem as we consider the decoupled method. Recalling (3.5.6) and (3.5.9) it is clear that $\|v_S\|_d\leq C_2\|v\|_d$ for some $C_2>0$. Hence

(3.5.13)
$$\tilde{\mathcal{B}}_d(v,\hat{v})=\tilde{\mathcal{B}}_d(v,v)+\gamma\tilde{\mathcal{B}}_d(v,v_S)\geq\frac{1}{4}\|v\|^2_d-\gamma C_1\|v\|_d\|v_S\|_d.$$

Thus if $\gamma<C_1C_2/8$ then $\tilde{\mathcal{B}}_d(v,\hat{v})\geq\frac{1}{8}\|v\|^2_d$. Combined with (3.5.11) we have (3.5.5).

□

3.6 Stability of the Decoupled and Weighted Approximations

In summary, we learned that under the above assumptions the decoupled approximation satisfies the stability bound:

$$\|\tilde{v}_h\|_{H^1(\mathcal{T}_{\mathrm{cG}})} \lesssim 1, \qquad \|\tilde{v}_h\|_S \lesssim \|f\|_{L^2(\Omega)}. \tag{3.6.1}$$

The first bound is a consequence of Lemmas 3.3.4 and 3.4.4, and the second of Theorem 3.5.1. So while we have streamline-diffusion stability on $\mathcal{T}_{\mathrm{dG}}$, an even stronger bound is available on $\mathcal{T}_{\mathrm{cG}}$. This finding is quite intuitive given that we expect the solution to have higher regularity in this region also.

Theorem 3.6.2. *Suppose that the operator norm of the trace $H^1(\mathcal{T}_{cG}) \to L^2(J)$ is bounded independently of h. Then the weighted cdG approximation $\overline{v}_h$ is stable in the sense that*

$$h_{\mathcal{T}_{cG}}\|\nabla_h \overline{v}_h\|^2_{L^2(\mathcal{T}_{cG})} + \|\overline{v}_h\|^2_S \lesssim 1 + \|f\|^2_{L^2(\Omega)}.$$

Proof. Set $\zeta := \overline{v}_h - \tilde{v}_h$. Using the coercivity of $\overline{\mathcal{B}}_\varepsilon$, Galerkin orthogonality and the norm of the trace $H^1(\mathcal{T}_{\mathrm{cG}}) \to L^2(J)$, we have

$$\begin{aligned}
\frac{1}{4}|||\zeta|||^2 &\le \overline{\mathcal{B}}_\varepsilon(\zeta,\zeta) = \tilde{\mathcal{B}}_\varepsilon(\tilde{v}_h,\zeta) - \overline{\mathcal{B}}_\varepsilon(\tilde{v}_h,\zeta) + \overline{\mathcal{B}}_\varepsilon(\overline{v}_h,\zeta) - \tilde{\mathcal{B}}_\varepsilon(\tilde{v}_h,\zeta) \\
&= \tilde{\mathcal{B}}_\varepsilon(\tilde{v}_h,\zeta) - \overline{\mathcal{B}}_\varepsilon(\tilde{v}_h,\zeta) \\
&= \sum_{e\in J} \varepsilon \int_e \nabla_h \tilde{v}_h|_{\mathcal{T}_{\mathrm{cG}}} \cdot [\![\zeta]\!] + \nabla_h \zeta|_{\mathcal{T}_{\mathrm{cG}}} \cdot [\![\tilde{v}_h]\!] - \sigma [\![\tilde{v}_h]\!] \cdot [\![\zeta]\!] \,\mathrm{d}s \\
&\lesssim \left(\varepsilon\|\nabla_h \tilde{v}_h\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \varepsilon\sigma\|[\![\tilde{v}_h]\!]\|^2_{L^2(J)}\right)^{1/2} \left(\varepsilon\|\nabla_h \zeta\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \varepsilon\sigma\|[\![\zeta]\!]\|^2_{L^2(J)}\right)^{1/2}
\end{aligned}$$

and thus

$$|||\zeta|||^2 \lesssim \varepsilon\|\nabla_h \tilde{v}_h\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \varepsilon\sigma\|[\![\tilde{v}_h]\!]\|^2_{L^2(J)}. \tag{3.6.3}$$

Dividing through by $h_{\mathcal{T}_{\mathrm{cG}}}$ and using an inverse inequality on $\rho\|\zeta\|_{L^2(E)}$ gives

$$h_{\mathcal{T}_{\mathrm{cG}}}\|\nabla_h\zeta\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} \lesssim h^{-1}_{\mathcal{T}_{\mathrm{cG}}}|||\zeta|||^2 \lesssim \frac{\varepsilon}{h_{\mathcal{T}_{\mathrm{cG}}}}\|\nabla_h\tilde{v}_h\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} + \frac{\varepsilon\sigma}{h_{\mathcal{T}_{\mathrm{cG}}}}\|[\![\tilde{v}_h]\!]\|^2_{L^2(J)}. \tag{3.6.4}$$

With Assumptions 2.3.9 and 3.2.17 as well as (3.6.1) we bound each of the terms in (3.6.4). Using a triangle inequality on $\|\nabla_h\zeta\|_{L^2(\mathcal{T}_{\mathrm{cG}})}$ we conclude that

$$h_{\mathcal{T}_{\mathrm{cG}}}\|\nabla_h\overline{v}_h\|^2_{L^2(\mathcal{T}_{\mathrm{cG}})} \lesssim 1.$$

To show that $\|\overline{v}_h\|_S$ is bounded we establish an inf-sup condition for $\overline{\mathcal{B}}_\varepsilon$. Indeed, (3.5.4) may be used without change. It remains to transfer (3.5.5) to $\overline{\mathcal{B}}_\varepsilon$. The inequality (3.5.11) is still available as the discretization is the same for both forms. However we now use $\overline{\mathcal{B}}_d(v,\hat{v}) \leq C_1|||v|||\cdot|||\hat{v}|||$ in place of (3.5.12), justified by Assumption 3.2.6. By using (3.5.6)-(3.5.9), we have $|||v_S||| \leq C_2|||v|||$ for some $C_2 > 0$. Hence

$$\overline{\mathcal{B}}_d(v,\hat{v}) = \overline{\mathcal{B}}_d(v,v) + \gamma\overline{\mathcal{B}}_d(v,v_S) \geq \frac{1}{4}\|v\|^2_d - \gamma C_1|||v|||||v_S|||. \tag{3.6.5}$$

For γC_1C_2 small enough and λ sufficiently large, $\gamma C_1|||v|||||v_S|||$ is bounded by $\frac{1}{8}\|v\|^2_d + \frac{1}{2}\mathcal{B}_{ar}(v,\hat{v})$, using again the positivity of the terms in (3.5.11). □

Observe that due to Assumption 3.2.17 the effect of the weaker elliptic penalisation on J is compensated for by the jumps of the first-order terms in the considered parameter regime.

3.7 Numerical Experiments

Example 3.7.1 *Let $\Omega = (0,1)^2$. We seek to solve*

$$-\varepsilon\Delta u + (-x,-y)\cdot\nabla u = -x - y \tag{3.7.1}$$

with Dirichlet boundary conditions chosen such that the solution is given by

$$u(x,y) = x + y - \frac{\mathrm{Erf}\left(\frac{x}{\sqrt{2\varepsilon}}\right) + \mathrm{Erf}\left(\frac{y}{\sqrt{2\varepsilon}}\right)}{\mathrm{Erf}\left(\frac{1}{\sqrt{2\varepsilon}}\right)} \tag{3.7.2}$$

where Erf *is the error function defined by*

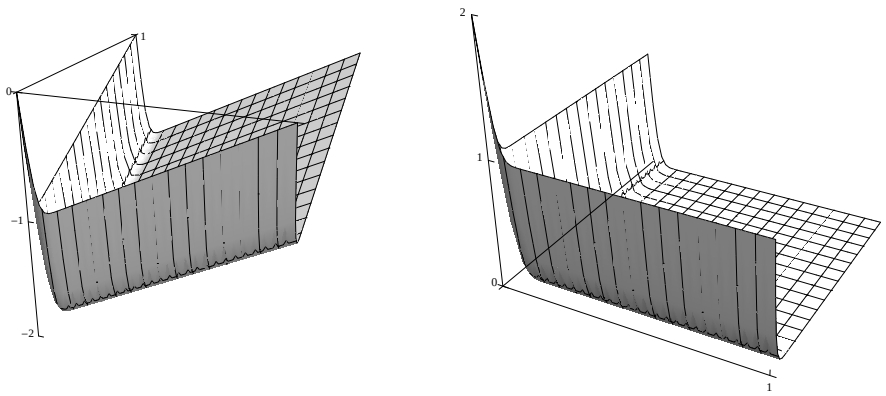
$$\mathrm{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-t^2}\,\mathrm{d}t.$$

For $0 < \varepsilon \ll 1$ *this problem exhibits an exponential boundary layer along the outflow boundaries* $x = 0$ *and* $y = 0$ *of width* $\mathcal{O}(\sqrt{\varepsilon})$.

Away from the layers the boundary conditions on the inflow boundaries $x = 1$ and $y = 1$ are well approximated by $y - 1$ and $x - 1$ respectively. The hyperbolic solution with these boundary conditions is given by $u_0(x, y) = x + y - 2$. This gives

$$u_\varepsilon(x,y) = 2 - \frac{\mathrm{Erf}\left(\frac{x}{\sqrt{2\varepsilon}}\right) + \mathrm{Erf}\left(\frac{y}{\sqrt{2\varepsilon}}\right)}{\mathrm{Erf}\left(\frac{1}{\sqrt{2\varepsilon}}\right)}. \tag{3.7.3}$$

We plot (3.7.2) and (3.7.3) for $\varepsilon = 10^{-3}$ in Figure 3.7.1. It is clear that away from the layers at the outflow boundaries the solution u_ε is close to zero.



(a) Solution u given by (3.7.2). (b) Solution u_ε given by (3.7.3)

Figure 3.7.1: Example 3.7.1 solution u and u_ε for $\varepsilon = 10^{-3}$. Away from the layers u_ε is very close to zero.

We attempt to identify Ω_{cG} by plotting $\|u_\varepsilon\|_{H^2(D)}$ on a region $D = (1 - \delta, 1)^2$,

$0 \leq \delta \leq 1$. If for a given D we have $\|u_\varepsilon\|_{H^2(D)} \lesssim 1$ for all $\varepsilon \leq \varepsilon_{\max}$ for some $\varepsilon_{\max}$, this D is an approximation for Ω_{cG}. As we can see from Figure 3.7.2 smaller $\varepsilon_{\max}$ allow for larger continuous regions.

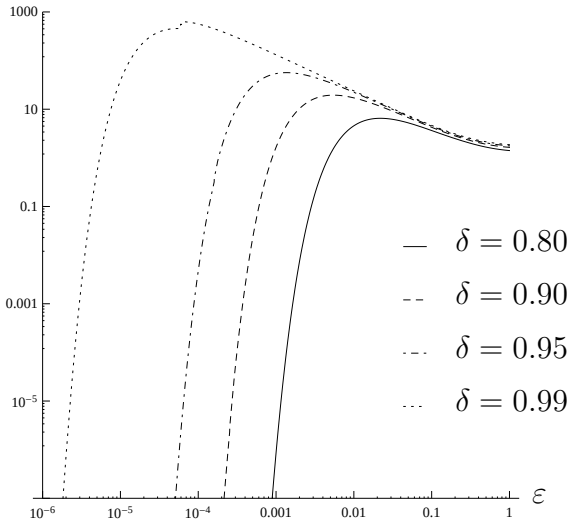


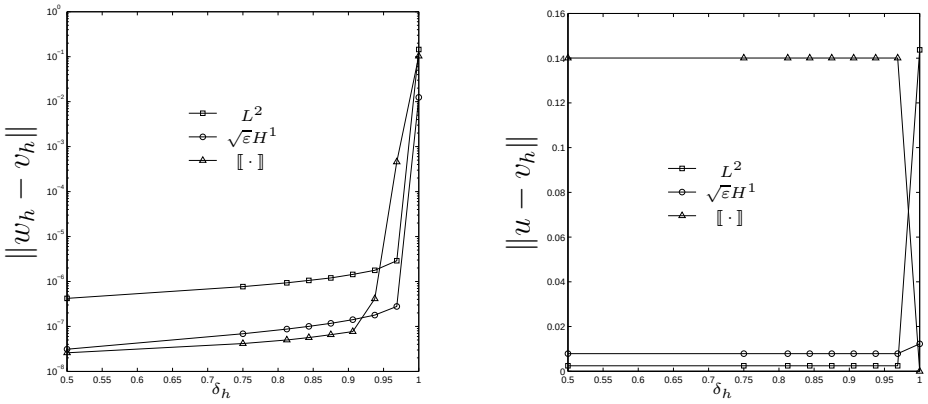
Figure 3.7.2: Example 3.7.1 plotting $\|u_\varepsilon\|_{H^2(D)}$ for various domains.

For this example $c - {}^1\!/_2\nabla \cdot b = 1$, so Assumption 2.1.5 is satisfied. Consider $\varepsilon_{\max} = 10^{-6}$ and a uniform mesh of quadrilaterals of edge length 2^{-5}. We will use piecewise bilinear elements. Then the smallest value of $\|b\|_{L^\infty(E)}$ for $E \in \mathcal{T}_h$ is 2^{-5}. Therefore Assumption 2.3.9 is satisfied, the smallest local mesh Péclet number being 488.28.

We define $\mathcal{T}_{\mathrm{cG}} = [1-\delta_h, 1]^2$, where $\delta_h = n2^{-5}$, $n \in \{0, 1, \ldots, 32\}$. Note that δ_h is discrete whereas δ is continuous (it is not possible to have half a cell in $\mathcal{T}_{\mathrm{cG}}$). The interface J is composed of the edges lying on the lines $y = \delta_h$ for $x \geq \delta_h$ and $x = \delta_h$ for $y \geq \delta_h$. The smallest value of $b \cdot n$ is δ_h occurring on the edges containing the point (δ_h, δ_h) and so Assumption 3.2.17 is satisfied for all possible $\mathcal{T}_h$ decompositions for this choice of $\varepsilon_{\max}$, h and $\sigma = 10$. From Figure 3.7.2 we can see that with $\mathcal{T}_{\mathrm{cG}}$ as defined we have $\mathcal{T}_{\mathrm{cG}} \subset \Omega_{\mathrm{cG}}$ for $\varepsilon_{\max} = 10^{-6}$.

In Figure 3.7.3 we plot the $L^2(\mathcal{T}_h)$ norm, $\sqrt{\varepsilon}$ weighted $H^1(\mathcal{T}_h)$ semi-norm and L^2 norm of the jumps on $\mathcal{E}_h$ (represented by $[\![\cdot]\!]$) for both the difference between the dG and cdG approximations and the error in the cdG approximation. In Figure 3.7.3(a) we see that the difference in the approximations increases only very slowly

until the final data point (where $\mathcal{T}_{\mathrm{cG}} = \mathcal{T}_h$). When the continuous region covers the layer non-physical oscillations pollute the approximation. The behaviour is even more marked when considering the error in Figure 3.7.3(b). Note that in this plot the jump terms have been scaled by a factor of 1/10.



(a) Difference between cdG and dG approximations.

(b) Error in the cdG approximation.

Figure 3.7.3: Comparing approximations for Example 3.7.1 for $\varepsilon = 10^{-6}$. The change as $\mathcal{T}_{\mathrm{cG}}$ covers the layer is apparent, with a large increase in the norms.

In Table 3.7.1 we show the number of degrees of freedom (dofs) as the continuous region is increased. Reducing the degrees of freedom to approximately 30% of the dG method results in only a very slight difference in the norm. For comparison with Becker et. al. [26] where $2n$ degrees of freedom are required here we have $1.17n$ with one row of elements in $\mathcal{T}_{\mathrm{dG}}$ (allowing for boundary conditions).

$1-\delta$	dofs	% of dG dofs	$\sqrt{\varepsilon}\|\nabla_h(w_h - v_h)\|_{H^1(\Omega)}$
dG	4096	100	0.0
8×2^{-5}	3361	82.1	3.1157e-08
16×2^{-5}	2417	59.0	6.8911e-08
24×2^{-5}	2121	51.8	8.7544e-08
30×2^{-5}	1457	35.6	1.7934e-07
31×2^{-5}	1276	31.2	2.7896e-07
cG	1089	26.6	1.2444e-02

Table 3.7.1: Degrees of freedom for Example 3.7.1 with $\varepsilon = 10^{-6}$. A considerable saving in degrees of freedom can be made without significantly increasing the difference in performance between the cdG and dG approximations.

We finally remark for this example that the choice of $\mathcal{T}_{\mathrm{cG}}$ leaving one layer of elements at the outflow boundary is in some sense optimum. If we add even one additional element we see the oscillations again pollute the region. For example, for

$$\mathcal{T}_{\mathrm{cG}} = [2^{-5}, 1]^2 \cup ([0.5, 0.5 + 2^{-5}] \times [0, 2^{-5}]),$$

i.e., adding a single element to $\mathcal{T}_{\mathrm{cG}}$ halfway along the x-axis, we find

$$\begin{aligned} \|w_h - v_h\|_{L^2(\Omega)} &= 4.7966 \times 10^{-2}, \\ \sqrt{\varepsilon}\|\nabla_h(w_h - v_h)\|_{L^2(\Omega)} &= 4.5008 \times 10^{-3}, \end{aligned}$$

a significant increase on the norms for $\mathcal{T}_{\mathrm{cG}} = [2^{-5}, 1]^2$. This choice of $\mathcal{T}_{\mathrm{cG}}$ also violates Assumption 3.2.17.

Example 3.7.2 *Let $\Omega = (0,1)^2$. We seek to solve*

$$-\varepsilon\Delta u + (1,1)\cdot\nabla u + u = f \tag{3.7.4}$$

with homogeneous Dirichlet boundary conditions and f chosen such that the solution is given by

$$u(x,y) = \left(x - \frac{\mathrm{e}^{(x-1)/\varepsilon} - \mathrm{e}^{-1/\varepsilon}}{1 - \mathrm{e}^{-1/\varepsilon}}\right)\left(y - \frac{\mathrm{e}^{(y-1)/\varepsilon} - \mathrm{e}^{-1/\varepsilon}}{1 - \mathrm{e}^{-1/\varepsilon}}\right). \tag{3.7.5}$$

For $0 < \varepsilon \ll 1$ this problem exhibits an exponential boundary layer along the outflow boundaries $x = 1$ and $y = 1$ of width $\mathcal{O}(\varepsilon)$.

For this example f depends on ε. However $f \to x + y + xy$ as $\varepsilon \to 0$ and the solution to the hyperbolic problem with zero boundary conditions in the limit is $u_0(x,y) = xy$. This u_0 is a good approximation to the hyperbolic solution for small ε away from the layers. We identify Ω_{cG} using u_ε given the limit solution u_0 and u defined above. Define $D = [0, \delta]$, $0 \le \delta \le 1$. In Figure 3.7.4 we plot $\|u_\varepsilon\|_{H^2(\Omega)}$ for various domains D. Note that compared to Example 3.7.1 the layer is sharper for a given ε. Therefore we see that for a given domain we may choose a larger $\varepsilon_{\max}$, or conversely for a given $\varepsilon_{\max}$ the region Ω_{cG} is larger compared to Example 3.7.1.

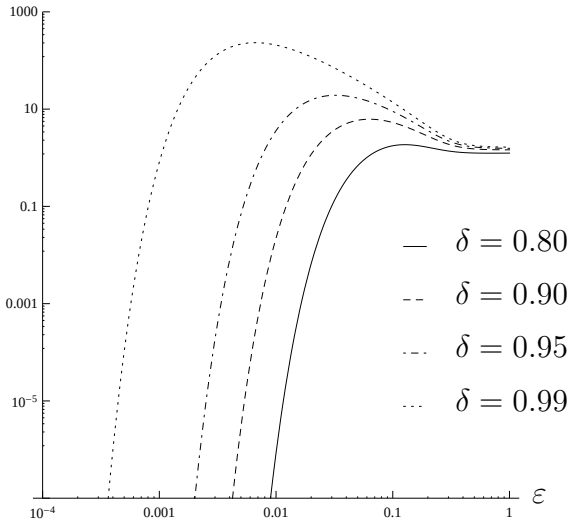
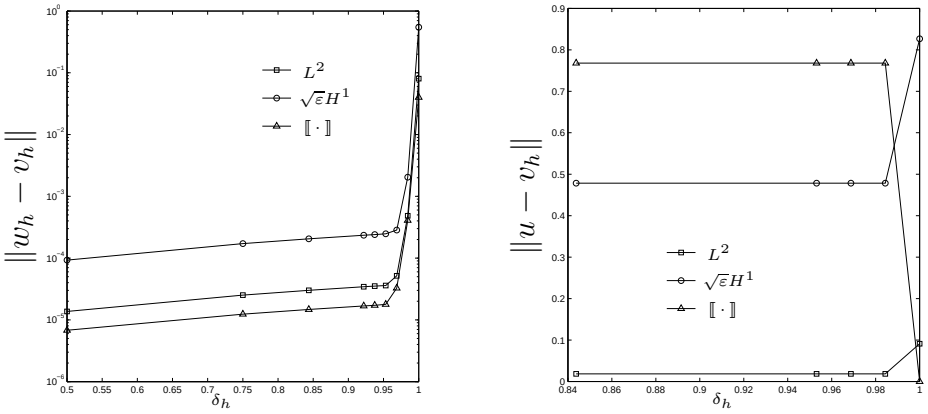


Figure 3.7.4: Example 3.7.2 plotting $\|u_\varepsilon\|_{H^2(D)}$ for various domains.

For this example $c - \frac{1}{2}\nabla \cdot b = 1$, so Assumption 2.1.5 is satisfied. Consider $\varepsilon_{\max} = 10^{-3}$ and a uniform mesh of quadrilaterals of edge length 2^{-6} and we again use piecewise bilinear elements. The value of $\|b\|_{L^\infty(E)}$ is fixed at 1 and so Assumption 2.3.9 is satisfied with the local mesh Péclet number being 7.8125 for every $E \in \mathcal{T}_h$.

We define $\mathcal{T}_{\mathrm{cG}} = [0, \delta_h]^2$, where $\delta_h = n2^{-6}$, $n \in \{0, 1, \ldots, 64\}$. The interface J is composed of the edges lying along the lines $y = 1 - \delta_h$ for $x \leq 1 - \delta_h$ and similarly with the x and y interchanged. As b is fixed we see that Assumption 3.2.17 is satisfied for all possible J with this choice of $\varepsilon_{\max}$ and h with $\sigma = 1$ provided $\mathcal{T}_{\mathrm{cG}} \subset \Omega_{\mathrm{cG}}$.

In Figure 3.7.5 we plot the $L^2(\mathcal{T}_h)$ norm, $\sqrt{\varepsilon}$ weighted $H^1(\mathcal{T}_h)$ semi-norm and L^2 norm of the jumps on $\mathcal{E}_h$ for both the difference of the dG and cdG approximations and the error in the cdG approximation. We see that there is little increase in the norms for the difference in the approximations until the final two data points (Figure 3.7.5(a)). Looking at Figure 3.7.4 we see that $\varepsilon = 10^{-3}$ gives an approximation to Ω_{cG} of $(0, 0.99)^2$. For the continuous region covering all but the final row of cells we have $\mathcal{T}_{\mathrm{cG}} = [0, 0.984]^2$, so the presence of some oscillations is as predicted. However the increase is not large enough to register on a plot of the norms of the error (Figure 3.7.5(b)). Covering the layer with Ω_{cG} entirely produces the expected large increase in the difference and the error.



(a) Difference between cdG and dG approximations.

(b) Error in the cdG approximation.

Figure 3.7.5: Comparing approximations for Example 3.7.2 for $\varepsilon = 10^{-3}$. The change as $\mathcal{T}_{\text{cG}}$ covers the layer is apparent, with the final two data points showing an increase in (a) and the final set showing an increase for (b).

In Figure 3.7.6 we investigate this difference further by plotting the error for the fixed decomposition with $\mathcal{T}_{\text{cG}} = [0, 1 - 2^{-6}]^2$ as we decrease ε. We now plot the unscaled $H^1(\mathcal{T}_h)$ semi-norm with the L^2 and L^2 jump norms. For $\varepsilon = 10^{-1}$ to 10^{-3} this $\mathcal{T}_{\text{cG}}$ partially covers the layer. When $\varepsilon = 10^{-1}$ the refinement of the mesh is sufficient to resolve the layer, and we see an increase in the $H^1(\mathcal{T}_h)$ semi-norm between $\varepsilon = 10^{-1}$ and 10^{-3} as we increasingly fail to resolve the layer but do not contain the layer in $\mathcal{T}_{\text{dG}}$. As the layer sharpens as ε decreases further the $L^2(\mathcal{T}_h)$ and $H^1(\mathcal{T}_h)$ norms decrease. The layer is not resolved but is entirely contained in $\mathcal{T}_{\text{dG}}$. For the jumps we see different behaviour. The largest jump is at the outflow boundary which is always in $\mathcal{T}_{\text{dG}}$. For relatively large ε this choice of $\mathcal{T}_{\text{cG}}$ removes some jumps away from the layer which would be present in the dG solution. However this is at the expense of some stability as can be seen from the $H^1(\mathcal{T}_h)$ norm.

We look at the savings made in degrees of freedom in Table 3.7.2 for $\varepsilon = 10^{-3}$. A reduction to approximately 30% of the degrees of freedom required for the dG method on this mesh does not cause a large increase in the difference between dG and cdG approximations. We use $1.18n$ degrees of freedom for the cdG approximation with two rows of elements at the outflow boundary in $\mathcal{T}_{\text{dG}}$, compared to $4n$ for the dG method (allowing for boundary conditions) and $2n$ for the space enriched with

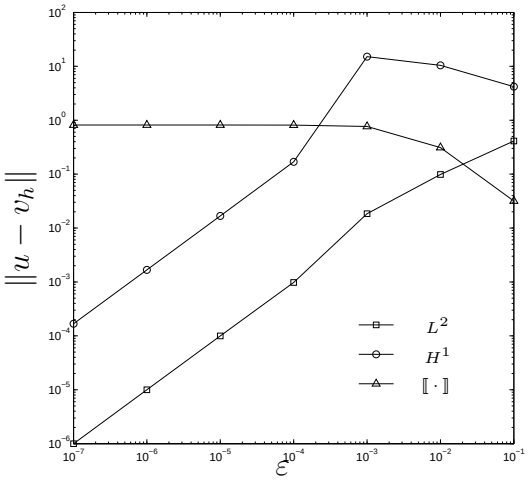


Figure 3.7.6: Reducing ε for a fixed $\mathcal{T}_{\mathrm{cG}} = [0, 1 - 2^{-6}]^2$ for Example 3.7.2. Once the layer is entirely outside of $\mathcal{T}_{\mathrm{cG}}$ the approximation improves.

piecewise constants as used in [26].

δ	dofs	% of dG dofs	$\sqrt{\varepsilon}\|\nabla_h(w_h - v_h)\|_{H^1(\Omega)}$
0×2^{-6}	16384	100	0.0
16×2^{-6}	13377	81.6	9.2781e-05
32×2^{-6}	9569	58.4	1.7153e-04
61×2^{-6}	5344	32.6	2.4733e-04
62×2^{-6}	4977	30.5	2.8491e-04
63×2^{-6}	4604	28.1	2.0398e-03
64×2^{-6}	4225	25.8	5.4685e-01

Table 3.7.2: Degrees of freedom for Example 3.7.2 with $\varepsilon = 10^{-3}$. A considerable saving in degrees of freedom can be made without significantly increasing the difference in performance between the cdG and dG approximations.

Part II

A Posteriori Error Estimators for Incompressible Miscible Displacement

Chapter 4

Introduction to the Equations of Incompressible Miscible Displacement

In this chapter we turn our attention to the equations of incompressible miscible displacement (1.1.3)-(1.1.8) and introduce preparatory material for further discussion of a posteriori error estimators in Chapters 5 and 6.

4.1 Literature Review

The existence and uniqueness of solutions to (1.1.3)-(1.1.8) was shown by Chen and Ewing in 1999 [52], but the numerical approximation of such equations has been discussed in the literature for considerably longer. We will not discuss anything prior to the work of Peaceman [110], but direct the reader to that text and the references therein which concern primarily a finite difference approach. For the finite element method the first appearance in the literature is [70], developing an a priori estimator for the pressure and concentration components.

Equations (1.1.4)-(1.1.5) can be shown to have a solution $(u, p) \in H(\mathrm{div}; \Omega) \times L^2(\Omega)$. We refer to numerical procedures to find approximations $(u_h, p_h) \in U \times P$, where U and P are finite dimensional subspaces of $H(\mathrm{div}; \Omega)$ and $L^2(\Omega)$ respectively, as *mixed* methods. See the book by Brezzi and Fortin [35] for a comprehensive

review. For the coupled problem of incompressible miscible displacement we refer to, e.g., mixed-continuous methods where a mixed method is used to approximate pressure and velocity and a continuous Galerkin method is used to approximate concentration.

An approach using a mixed-continuous Galerkin method to solve for pressure, velocity and concentration was introduced in [64] and the pressure-velocity components were further studied in [65]. For a more complete history see the references in [123] in which the mixed-discontinuous Galerkin scheme was introduced and a priori results generated through the use of a cut off functional. Further results have been shown a priori for the compressible case [56, 57]. Extensions to the case of minimal regularity have been addressed in [20] and a Crank-Nicolson solution scheme for the mixed-dG case proposed in [85].

For an overview of residual a posteriori error estimation see the book by Ainsworth and Oden, [5]. Reliable a posteriori indicators for the mixed-cG case were introduced in [51] and a dG-dG method for the compressible problem in [128]. For the case of uncoupled Darcy flow approximated using a dG method, [19] proposes a reliable and efficient estimator. By *reliable* we refer to an estimator that gives an upper bound on the error. Estimators which give a lower bound on the error are referred to as *efficient*, but we will not consider estimators of this type in this thesis. There are also results using the dual weighted residual method [27] in the case of one way coupling [95] where the refinement is formulated to achieve a particular numerical goal. To our knowledge no papers have been published concerning goal oriented adaptivity with two way coupling.

There has also been considerable contribution to this field from Wheeler and her collaborators. In [123] the authors present an optimal a priori error estimate (in $L^2((0,T];H^1(\Omega))$ for concentration and $L^\infty((0,T];L^2(\Omega))$ for velocity) for a mixed-dG scheme. For a dG-dG scheme Sun and Wheeler [124] present a priori error estimates for a system of coupled equations that also include reaction terms. Sun and Wheeler [125, 126] also consider a posteriori error estimators for dG approximations to the reactive transport problem, i.e., similar to (1.1.3) with an additional reaction term. Estimators in both $L^2((0,T];H^1(\Omega))$ [125] and $L^2((0,T];L^2(\Omega))$ [126] are

shown, and the authors remark that estimators of the second type are preferred for the concentration.

4.2 The Coefficients of the Problem

We make the following assumptions on the coefficients of the problem (1.1.3)-(1.1.5) (cf., [20, 52]):

(A1) We have $\mathbb{K} \in L^\infty(\Omega; \mathbb{R}^{d\times d})$ and there exists positive real numbers $k_\circ, k^\circ$ such that

$$k_\circ |\xi|^2 \leq \xi^\top \mathbb{K}(x)\xi \leq k^\circ |\xi|^2$$

for all $x \in \Omega$ and $\xi \in \mathbb{R}^d$. Moreover, $\mathbb{K}(x)$ is symmetric;

(A2) There exist positive real numbers $\mu_\circ$, μ° such that the Lipschitz continuous function $\mu : \mathbb{R} \to \mathbb{R}$ satisfies $\mu_\circ \leq \mu(c) \leq \mu^\circ$ for all $c \in \mathbb{R}$;

(A3) There exist positive real numbers $d_\circ \leq 1 \leq d^\circ$ such that the function $\mathbb{D} : \mathbb{R}^d \times \Omega \to \mathbb{R}^{d\times d}$ satisfies the Carathéodory condition

$$\begin{aligned} &\mathbb{D}(u,\cdot) : x \mapsto \mathbb{D}(u,x) \quad \text{is measurable on } \Omega \text{ for all } u \in \mathbb{R}, \\ &\mathbb{D}(\cdot,x) : u \mapsto \mathbb{D}(u,x) \quad \text{is continuous on } \mathbb{R}^d \text{ for almost all } x \in \Omega \end{aligned}$$

and the two sided, u-dependent growth condition

$$d_\circ(1+|u|)|\xi|^2 \leq \xi^\top \mathbb{D}(u,x)\xi \leq d^\circ(1+|u|)|\xi|^2$$

for all $u, \xi \in \mathbb{R}^d$ and $x \in \Omega$. Furthermore, $\mathbb{D}(u,x)$ is symmetric for $(u,x) \in \mathbb{R}^d \times \Omega$;

(A4) We have $\varphi \in W^{2,\infty}(\Omega)$, and there are positive $\varphi_\circ, \varphi^\circ \in \mathbb{R}$ such that $\varphi_\circ \leq \varphi(x) \leq \varphi^\circ$;

(A5) We have $q^I, q^P \in L^\infty((0,T]; L^2(\Omega))$ satisfy $q^I, q^P \geq 0$ in Ω_T and

$$\int_\Omega q^I(t,x) - q^P(t,x)\,\mathrm{d}\boldsymbol{x} = 0$$

for $t \in (0,T]$;

(A6) We have $\hat{c} \in L^\infty((0,T]\times\Omega)$ and $c_0 \in L^\infty(\Omega)$ satisfy $0 \leq \hat{c}(t,x)$, $c_0(x) \leq 1$ in $(0,T]\times\Omega$ and Ω respectively; additionally we assume for simplicity that $c_0 \in V_{\mathrm{dG}}$.

(A7) There exist positive real numbers $\rho_\circ$, ρ° such that the Lipschitz continuous function $\rho : \mathbb{R} \to \mathbb{R}$ satisfies $\rho_\circ \leq \rho(c) \leq \rho^\circ$ for all $c \in \mathbb{R}$. Also g is a constant vector in $\mathbb{R}^d$.

We make the common specific choice for the diffusion dispersion tensor, e.g., [52, 71, 123]

$$\mathbb{D}(u,\boldsymbol{x}) = \varphi\left(d_m\mathbb{I} + |u|d_l\mathbb{E}(u) + |u|d_t(\mathbb{I}-\mathbb{E}(u))\right) \tag{4.2.1}$$

where $\mathbb{E}(u) = uu^\top/|u|^2$ and $\mathbb{I}$ is the identity matrix. We specify that the molecular, longitudinal and transverse diffusion coefficients d_m, d_l and d_t are positive real numbers.

4.3 Regularity

We will make frequent use of the regularity bounds for solutions of elliptic and parabolic equations in convex domains. We therefore present a general discussion of the regularity of such equations which can be found in, e.g., [69].

Consider a second order elliptic partial differential operator $\mathcal{L}$ given by

$$\mathcal{L}\psi = -\sum_{i,j=1}^{d}(a^{ij}(x)\psi_{x_i})_{x_j} + \sum_{i=1}^{d} b^i(x)\psi_{x_i} + c(x)\psi$$

where a^{ij}, b^i and c are given coefficient functions. Then we define the following

boundary value problem on Ω

$$\begin{aligned} \mathcal{L}\psi &= f \qquad \text{on } \Omega \\ \psi &= 0 \qquad \text{on } \partial\Omega \end{aligned} \tag{4.3.1}$$

where $f : \Omega \to \mathbb{R}$ is a known function. Then we have the following theorem ([69, Section 6.3]).

Theorem 4.3.2. *Assume $a^{ij} \in C^1(\overline{\Omega})$, $b^i, c \in L^\infty(\Omega)$ and $f \in L^2(\Omega)$. Suppose ψ is the unique solution of (4.3.1). Then if $\partial\Omega$ is C^2 or the region is convex we have that $\psi \in H^2(\Omega)$ and the regularity bound*

$$\|\psi\|_{H^2(\Omega)} \leq C\|f\|_{L^2(\Omega)} \tag{4.3.3}$$

where C depends on Ω, a^{ij}, b^i and c.

If we have higher regularity in the coefficients and right hand side and domain it is possible to show that the solution ψ also has higher regularity.

Suppose now we look at the parabolic problem

$$\begin{aligned} \frac{\partial\zeta}{\partial t} + \mathcal{L}\zeta &= f \qquad \text{on } \Omega_T \\ \zeta &= 0 \qquad \text{on } \partial\Omega_T \\ \zeta(0,x) &= g \qquad \text{on } \Omega. \end{aligned} \tag{4.3.4}$$

Then we have the following regularity theorem, e.g., [69, Section 7.1.3]:

Theorem 4.3.5. *Assume $a^{ij} \in L^2((0,T]; C^1(\overline{\Omega}))$, $b^i, c \in L^2((0,T]; L^\infty(\Omega))$ and $f \in L^2((0,T]; L^2(\Omega))$. Also assume $g \in H_0^1(\Omega)$, i.e., those functions in H^1 with zero trace on the boundary $\partial\Omega$. Then for $\zeta \in L^2((0,T]; H_0^1(\Omega))$ the weak solution of (4.3.4) we have the estimate*

$$\operatorname*{ess\,sup}_{t\in(0,T]} \|\zeta(t)\|_{H_0^1(\Omega)} + \|\zeta\|_{L^2((0,T];H^2(\Omega))} \leq C\left(\|f\|_{L^2((0,T];L^2(\Omega))} + \|g\|_{H_0^1(\Omega)}\right) \tag{4.3.6}$$

where C depends on Ω, a^{ij}, b^i, c and T.

With the Sobolev embedding Theorems [3] (see also Chapter 6) this means that in two or three dimensions we have $p, c \in L^\infty(\Omega)$. For convex domains we have additionally that $u \in L^\infty(\Omega)$ (see, e.g., [102, Chapter 7] for a full discussion), which does not hold for non convex domains see, e.g., [77]. We will seek an alternative approach in regions where $u \notin L^\infty(\Omega)$ in Chapter 6 through the application of weighted spaces.

4.4 The Continuous Time Raviart-Thomas dG Finite Element Method

Define

$$L_0^2(\Omega) := \left\{ w \in L^2(\Omega) : \int_\Omega w \, \mathrm{d}\boldsymbol{x} = 0 \right\}. \tag{4.4.1}$$

The solution to the flow problem for pressure is unique only up to an addative constant, which we determine by specifying $p \in L_0^2(\Omega)$. We assume the solution is smooth enough so the weak form of (1.1.3)-(1.1.5) is given by: Find $(u, p, c) \in L^\infty((0,T]; H_0(\mathrm{div}; \Omega)) \times L^\infty((0,T]; L_0^2(\Omega)) \times L^2((0,T]; H^{n-1}(\Omega))$ and $\partial c/\partial t \in L^2((0,T]; H^{n-1}(\Omega))$ such that

$$\begin{aligned} &\left(\varphi \frac{\partial c}{\partial t}, d\right) + \sum_{E \in \mathcal{T}_h} (\mathbb{D}(u)\nabla c, \nabla d)_E \\ &\quad + \sum_{E \in \mathcal{T}_h} \left[(u \cdot \nabla c, d)_E + (q^I c, d)_E \right] = (\hat{c} q^I, d) \end{aligned} \tag{4.4.2}$$

for all $d \in H^1(\Omega)$ and a.e. $t \in (0,T]$, and

$$(\nabla \cdot u, w) = (q^I - q^P, w) \tag{4.4.3}$$

$$(a^{-1}(c)u, v) - (p, \nabla \cdot v) = (\rho(c) g, v) \tag{4.4.4}$$

for all $(v, w) \in H(\mathrm{div}; \Omega) \times L_0^2(\Omega)$ and a.e. $t \in (0,T]$. The regularity of the solution is given by $n \geq 2$. Compare this to the discussion of the guaranteed regularity in the previous section, Assumption 5.1.11 and the discussion in Chapter 6.

We solve for the pressure and velocity using a Raviart-Thomas (RT) procedure [66, 113] and for the concentration using a dG method. We refer to the whole scheme as a RT-dG method. For polynomial degree $k \geq 0$ and restricting ourselves to two dimensions we define the global Raviart-Thomas finite element space by

$$\mathcal{RT}_k(\mathcal{T}_h) := \{v \in H(\mathrm{div};\Omega) : v|_E \in [\mathbb{P}^k(E)]^2 + \boldsymbol{x}\mathbb{P}^k(E)\ \forall E \in \mathcal{T}_h\}. \tag{4.4.5}$$

Recall the definition of $H_0(\mathrm{div};\Omega)$ from (1.3.4). Then u is approximated in the space

$$U := \mathcal{RT}_k(\mathcal{T}_h) \cap H_0(\mathrm{div};\Omega).$$

For the pressure we define the approximation space

$$P := V_{\mathrm{dG}} \cap L_0^2(\Omega)$$

where V_{dG} is defined in (2.2.2). Then the velocity and pressure are approximated in $U \times P$. Note that $U \times P \subset H_0(\mathrm{div};\Omega) \times L_0^2(\Omega)$. To simplify the presentation we use the same mesh $\mathcal{T}_h$ to solve for u, p and c numerically at a given time and stipulate there is no refinement of the polynomial degree.

For the diffusion part of the concentration equation define the bilinear form

$$\begin{aligned}\mathcal{B}_d(c_h, d_h; u_h) = &\sum_{E\in\mathcal{T}_h}\int_E \mathbb{D}(u_h)\nabla_h c_h \cdot \nabla_h d_h \,\mathrm{d}\boldsymbol{x} + \sum_{e\in\mathcal{E}_h^o}\int_e \sigma[\![c_h]\!]\cdot[\![d_h]\!]\,\mathrm{d}s \\ &- \sum_{e\in\mathcal{E}_h^o}\int_e [\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h)\nabla_h d_h\}\!\} + [\![d_h]\!]\cdot\{\!\{\mathbb{D}(u_h)\nabla_h c_h\}\!\}\,\mathrm{d}s\end{aligned} \tag{4.4.6}$$

for all $d_h \in V_{\mathrm{dG}}$. The penalty parameter σ is defined by [20]

$$\sigma : \mathcal{E}_h \to \mathbb{R}, \quad \boldsymbol{x} \mapsto C_{\mathrm{pen}} \frac{\max\{n_{\mathcal{E}_h}^\top \mathbb{D}(u_h^+, \boldsymbol{x}) n_{\mathcal{E}_h}, n_{\mathcal{E}_h}^\top \mathbb{D}(u_h^-, \boldsymbol{x}) n_{\mathcal{E}_h}\}}{h}$$

and C_{pen} is chosen such that it is larger than

$$\sup\left\{h \max\left\{\frac{\|\nu_h\|_{\partial E}^2}{\|\nu_h\|_E^2}, \frac{\|D^{1/2}\nabla_h\nu_h\|_{\partial E}^2}{\|D^{1/2}\nabla_h\nu_h\|_E^2}\right\} : \nu_h \in \mathbb{P}^s, D \in [\mathbb{P}^s]^{d\times d}\right\}. \tag{4.4.7}$$

The bilinear form for convection, production and injection is given by (cf., (7.3.2))

$$\begin{aligned}\mathcal{B}_{cq}^{\text{alt}}(c_h, d_h; u_h) = &\sum_{E\in\mathcal{T}_h}\int_E (u_h\cdot\nabla_h c_h)d_h + (q^I c_h)d_h \,\mathrm{d}\boldsymbol{x} \\ &- \sum_{e\in\mathcal{E}_h^o}\int_e (u_h\cdot[\![c_h]\!])d_h^* \,\mathrm{d}s\end{aligned} \tag{4.4.8}$$

where d_h^* is defined by

$$d_h^* = \begin{cases} d_h^- & \text{if } u_h\cdot n^+ > 0, \\ d_h^+ & \text{if } u_h\cdot n^+ \le 0. \end{cases} \tag{4.4.9}$$

Definition 4.4.10. *Define the continuous time RT-dG approximation* $(u_h, p_h, c_h) \in L^\infty((0,T];U)\times L^\infty((0,T];P)\times L^\infty((0,T];V_{dG})$ *to* (1.1.3)-(1.1.8) *as that which satisfies*

$$\left(\varphi\frac{\partial c_h}{\partial t}, d_h\right) + \mathcal{B}_d(c_h, d_h; u_h) + \mathcal{B}_{cq}^{\text{alt}}(c_h, d_h; u_h) = (\hat{c}q^I, d_h) \tag{4.4.11}$$

for all $d_h \in V_{dG}$ *and* $t\in(0,T]$,

$$(\nabla\cdot u_h, w_h) = (q^I - q^P, w_h), \tag{4.4.12}$$

$$(a^{-1}(c_h)u_h, v_h) - (p_h, \nabla\cdot v_h) = (\rho(c_h)g, v_h) \tag{4.4.13}$$

for all $(v_h, w_h)\in U\times P$ *and* $t\in(0,T]$, *and*

$$(c_h, d_h) = (c_0, d_h) \tag{4.4.14}$$

for all $d_h\in V_{dG}$ *and* $t = 0$.

Note that the equations (1.1.6)-(1.1.7) are satisfied for the approximation through the definition of the RT space, i.e., by construction $U\subset H_0(\text{div};\Omega)$ so (1.1.6) is satisfied and the choice of the diffusion dispersion tensor (4.2.1) ensures (1.1.7) is likewise satisfied.

Chapter 5

A Posteriori Error Estimators for the Raviart-Thomas dG Finite Element Method

In this chapter we introduce an a posteriori error estimator for the Raviart Thomas dG finite element method as presented in Definition 4.4.10. To our knowledge no estimator for this scheme exists in the literature. We however remark that Chen and Liu have shown an estimator for the RT-cG scheme (using a different approach to coupling the estimates), and Yang [128] has presented an estimate for a dG-dG scheme for the compressible problem. Our analysis finds inspiration in these papers but is significantly different. We also acknowledge that a posteriori estimators exist for the uncoupled equations: For the velocity pressure components [48, 66]; and for the concentration components [126]. Our analysis presents these estimates with coupling and formulates a combined error estimator.

5.1 Notation and Preliminary Results

Define the error terms by $\mathtt{E_u} := u - u_h$, $\mathtt{E_p} := p - p_h$ and $\mathtt{E_c} := c - c_h$.

We present the following construction of the Raviart-Thomas projection from [66]. For an element $E \in \mathbb{R}^d$ the local Raviart-Thomas space of order $k \geq 0$ is

defined by

$$\mathcal{RT}_k(E) = [\mathbb{P}^k(E)]^d + \boldsymbol{x}\mathbb{P}^k(E).$$

Then define the local interpolation operator $\Pi_E : [H^1(E)]^d \to \mathcal{RT}_k(E)$ via the following lemma [66, Lemma 3.2].

Lemma 5.1.1. *Given $v \in [H^1(E)]^d$ there exists a unique $\Pi_E v \in \mathcal{RT}_k(E)$ such that for each edge e of E we have*

$$\int_e \Pi_E v \cdot n \; p \, \mathrm{d}s = \int_e v \cdot n \; p \, \mathrm{d}s \qquad \forall p \in \mathbb{P}^k(e)$$

and if $k \geq 1$

$$\int_E \Pi_E v \cdot p \, \mathrm{d}\boldsymbol{x} = \int_E v \cdot p \, \mathrm{d}\boldsymbol{x} \qquad \forall p \in [\mathbb{P}^k(E)]^d.$$

Then we denote the global Raviart-Thomas projection

$$\Pi_h^{\mathcal{RT}} : H(\mathrm{div}; \Omega) \cap \prod_{E \in \mathcal{T}_h} [H^1(E)]^d \to \mathcal{RT}_k(\mathcal{T}_h)$$

defined by

$$\Pi_h^{\mathcal{RT}} v|_E = \Pi_E v \qquad \forall E \in \mathcal{T}_h.$$

This projection satisfies (see [66, Lemma 3.5])

$$\left(\nabla \cdot (v - \Pi_h^{\mathcal{RT}}(v)), w_h\right) = 0 \qquad \forall w_h \in V_{\mathrm{dG}}^k \tag{5.1.2}$$

and

$$\|v - \Pi_h^{\mathcal{RT}}(v)\|_{[L^2(\Omega)]^d} \leq C h^m \|v\|_{[H^m(\Omega)]^d} \quad 1 \leq m \leq k+1 \tag{5.1.3}$$

where C depends on the regularity of the mesh and k. Note that the local Raviart-Thomas projection satisfies a similar inequality, cf., [66, Theorem 3.1] which can be used to show (5.1.3).

We introduce the Clément interpolation operator $\mathcal{C}l_h(\,\cdot\,)$ [53]. See also [67, Section 1.6.1]. This is an operator $L^1(\Omega) \ni v \to \mathcal{C}l_h(v) \in V_{\mathrm{cG}}$, where V_{cG} is the degree r piecewise continuous polynomial approximation space defined in (2.1.8). The

Clément operator has the following interpolation properties [67]:

Lemma 5.1.4 (Clément Interpolation)**.** *There is a C such that for all $0 \leq m \leq 1$*

$$\|\mathcal{C}l_h(v)\|_{H^m(\Omega)} \leq C\|v\|_{H^m(\Omega)} \quad \forall v \in H^m(\Omega). \tag{5.1.5}$$

If $0 \leq m \leq l \leq r+1$ then for all h and all $E \in \mathcal{T}_h$ we have the approximation

$$\|v - \mathcal{C}l_h(v)\|_{H^m(E)} \leq Ch_E^{l-m}|v|_{H^l(\Delta_E)} \quad \forall v \in H^l(\Delta_E) \tag{5.1.6}$$

where Δ_E is the set of elements in $\mathcal{T}_h$ sharing at least one vertex with E. If $m+1/2 \leq l \leq r+1$ then for all h and all $e \in \mathcal{E}_h$ we have the approximation

$$\|v - \mathcal{C}l_h(v)\|_{H^m(e)} \leq Ch_e^{l-m-1/2}|v|_{H^l(\Delta_e)} \quad \forall v \in H^l(\Delta_e) \tag{5.1.7}$$

where Δ_e is the set of elements in $\mathcal{T}_h$ sharing at least one vertex with e.

We present the following lemma from, e.g., [124, Lemma 5.2].

Lemma 5.1.8. *With the assumption (A3) and $d_l, d_m, d_t \geq 0$, d_l, d_t bounded and the definition of $\mathbb{D}(u, \boldsymbol{x})$ from (4.2.1) we have*

$$\sum_{E\in\mathcal{T}_h} \|\mathbb{D}(u) - \mathbb{D}(v)\|_{L^2(E)} \leq C \sum_{E\in\mathcal{T}_h} \|u - v\|_{L^2(E)} \tag{5.1.9}$$

for all $u, v \in [L^\infty(\Omega)]^d$ where C is a fixed constant depending only on the bounds of d_t and d_l, and the dimension.

We have the following approximation properties see, e.g., [11, 116]. For $E \in \mathcal{T}_h$ and $\Psi \in H^\lambda(\mathcal{T}_h)$ we can find $\Psi_I \in \mathbb{P}^r(E)$ such that there exists a constant C independent of h_E and Ψ (but dependent on r and λ) for $0 \leq i \leq \lambda$ so that

$$\|\Psi - \Psi_I\|_{H^i(E)} \leq Ch_E^{\lambda-i}\|\Psi\|_{H^\lambda(E)} \qquad \lambda \geq 0. \tag{5.1.10}$$

We make the following assumptions on the regularity of the solution to the IMD equations (1.1.3)-(1.1.8), cf., Section 4.3.

Assumption 5.1.11. *Assume that (u, p, c), the solution to (1.1.3)-(1.1.8), satisfies the following regularity requirements: For $s_p \geq 2$ and $s_c \geq 2$ we have $p \in L^2((0,T]; H^{s_p}(\mathcal{T}_h))$, $u \in [L^2((0,T]; H^{s_p-1}(\mathcal{T}_h))]^d$ and $c \in L^2((0,T]; H^{s_c}(\mathcal{T}_h))$. Additionally assume that $\partial c/\partial t \in L^2((0,T]; H^{s_c-1}(\mathcal{T}_h))$ and that the initial condition $c_0 \in V_{dG}$. Also assume that p, ∇p, c and ∇c are essentially bounded (and hence u is also essentially bounded).*

5.2 An A Posteriori Estimator for the Pressure and Velocity

We first present an estimator for the pressure and velocity in terms of $\mathtt{E_c}$ and known quantities. This is based on the presentation for the RT-cG method in [51] with extensions for the discontinuous concentrations and gravity terms.

To begin the analysis we consider the problem: Find $(\tilde{u}, \tilde{p}) \in H_0(\mathrm{div}; \Omega) \times L_0^2(\Omega)$ such that for a.e. $t \in (0,T]$

$$(\nabla \cdot \tilde{u}, w) = (q^I - q^P, w) \qquad \forall w \in L_0^2(\Omega), \tag{5.2.1}$$

$$(a^{-1}(c_h)\tilde{u}, v) - (\nabla \cdot v, \tilde{p}) = (\rho(c_h)g, v) \qquad \forall v \in H_0(\mathrm{div}; \Omega). \tag{5.2.2}$$

This is similar to the weak form for the pressure and velocity equations (4.4.3)-(4.4.4) but with the numerical approximation to concentration.

We now introduce the following dual problem with continuous coefficients:

$$\nabla \cdot \xi = \tilde{p} - p_h \qquad \text{on } \Omega_T, \tag{5.2.3}$$

$$\xi = -a(\mathcal{C}_h(c_h))\nabla\psi \qquad \text{on } \Omega_T, \tag{5.2.4}$$

$$\xi \cdot n = 0 \qquad \text{on } \partial\Omega_T. \tag{5.2.5}$$

As ψ is defined only up to an additive constant we specify $\psi \in L_0^2(\Omega)$. Then (5.2.3)-(5.2.5) admits the following estimate on convex domains, e.g., [77, Chapter 3]:

$$\|\psi\|_{H^2(\Omega)} \leq C\|\tilde{p} - p_h\|_{L^2(\Omega)}. \tag{5.2.6}$$

Recall $a^{-1}(c) := \mathbb{K}^{-1}\mu$, and so by (A1) and (A2) we have $\alpha_\circ \leq a^{-1} \leq \alpha^\circ$. From (5.2.6) we can show $\|\xi\|_{H^1(\Omega)} \leq (1/\alpha_\circ)\|\nabla\psi\|_{H^1(\Omega)} \leq C\|\tilde{p}-p_h\|_{L^2(\Omega)}$ where C depends on the bound of $a^{-1}(c_h)$.

We make the following assumption concerning the size of the difference between the approximation c_h and its Clément projection. As we expect the term to scale with h if this assumption is not met it would be necessary to refine the mesh globally. This should be taken into account during potential de-refinement in an adaptive method.

Assumption 5.2.7. *With the Clément projection defined in Section 5.1 and c_h the finite element approximation to concentration defined in Definition 4.4.10 we assume that the lower bound of a^{-1} satisfies $\alpha_\circ > C\|a^{-1}(c_h) - a^{-1}(\mathcal{C}l_h(c_h))\|_{L^\infty(\Omega)}$ where C is the regularity coefficient of $\|\xi\|_{L^2(E)} \leq C\|\tilde{p} - p_h\|_{L^2(\Omega)}$ derived from (5.2.6).*

Theorem 5.2.8. *Let the conditions of Assumptions 5.1.11 and 5.2.7 hold and let (u_h, p_h, c_h) be the approximation defined in Definition 4.4.10. Then there exist constants C_1 and C_2 depending perhaps on the constants of the Clément approximation in Lemma 5.1.4, a trace inequality, the regularity bound (4.3.3) (restated in (5.2.6)), the polynomial degree of the approximation space and the bounds in (A1), (A2) and (A7), but each independent of h, such that*

$$\begin{aligned} &\|\mathtt{E_u}\|^2_{L^2((0,T];L^2(\Omega))} + \|\mathtt{E_p}\|^2_{L^2((0,T];L^2(\Omega))} \\ &\leq \mathscr{E}_{up} + \frac{C_1}{\alpha_\circ^2}\left(\rho^\circ\|g\|_{L^\infty(\Omega)} + \alpha^\circ\|u\|_{L^\infty(\Omega)}\right)^2 \|\mathtt{E_c}\|^2_{L^2((0,T];L^2(\Omega))} \end{aligned} \tag{5.2.9}$$

where

$$\begin{aligned} \mathscr{E}_{up} := C_2 \sum_{E\in\mathcal{T}_h} \Big(&(1+h_E^2)\|a^{-1}(c_h) - \rho(c_h)g\|^2_{L^2((0,T];L^2(E))} \\ &+ (1+h_E^2+h_E^4)\|q^I - q^P - \nabla\cdot u_h\|^2_{L^2((0,T];L^2(E))}\Big) \end{aligned} \tag{5.2.10}$$

Proof. By subtracting (5.2.2) from (4.4.4) we have

$$\begin{aligned} &(a^{-1}(c_h)(u-\tilde{u}), v) - (\nabla\cdot v, p-\tilde{p}) \\ &= ((\rho(c)-\rho(c_h))g, v) - ((a^{-1}(c)-a^{-1}(c_h))u, v) \quad \forall v \in H_0(\mathrm{div};\Omega). \end{aligned}$$

By taking $v = u - \tilde{u} \in H_0(\mathrm{div};\Omega)$ and using (4.4.3) and (5.2.1) we find

$$\begin{aligned}\alpha_\circ \|u-\tilde{u}\|^2_{L^2(\Omega)} &\le (a^{-1}(c_h)(u-\tilde{u}), u-\tilde{u}) \\ &= (\nabla\cdot(u-\tilde{u}), p-\tilde{p}) + ((a^{-1}(c_h)-a^{-1}(c))u, u-\tilde{u}) \\ &\quad + ((\rho(c)-\rho(c_h))g, u-\tilde{u}) \\ &\le \left(\rho^\circ \|g\|_{L^\infty(\Omega)} + \alpha^\circ \|u\|_{L^\infty(\Omega)}\right) \|\mathtt{E_c}\|_{L^2(\Omega)} \|u-\tilde{u}\|_{L^2(\Omega)}.\end{aligned}$$

Using the boundedness of $a^{-1}(c)$ and a Poincaré inequality we have

$$\begin{aligned}\|u-\tilde{u}\|_{L^2(\Omega)} &\ge C\|\nabla(p-\tilde{p})\|_{L^2(\Omega)} \\ &\ge C\|p-\tilde{p}\|_{H^1(\Omega)} \\ &\ge C\left(\|p-\tilde{p}\|_{L^2(\Omega)} + \|u-\tilde{u}\|_{L^2(\Omega)}\right)\end{aligned}$$

and by combining these two results we have

$$\begin{aligned}\|p-\tilde{p}\|^2_{L^2(\Omega)} + \|u-\tilde{u}\|^2_{L^2(\Omega)} &\le C\|u-\tilde{u}\|^2_{L^2(\Omega)} \\ &\le \frac{C}{\alpha_\circ^2}\left(\rho^\circ \|g\|_{L^\infty(\Omega)} + \alpha^\circ \|u\|_{L^\infty(\Omega)}\right)^2 \|\mathtt{E_c}\|^2_{L^2(\Omega)}.\end{aligned} \tag{5.2.11}$$

By subtracting (4.4.12) and (4.4.13) from (5.2.1) and (5.2.2) we obtain the orthogonality relationships

$$(\nabla\cdot(\tilde{u}-u_h), w_h) = 0 \qquad \forall w_h \in P, \tag{5.2.12}$$

$$(a^{-1}(c_h)(\tilde{u}-u_h), v_h) - (\nabla\cdot v_h, \tilde{p}-p_h) = 0 \qquad \forall v_h \in U. \tag{5.2.13}$$

The variational problem for (5.2.3)-(5.2.4) is

$$\begin{aligned}(\nabla\cdot\xi, w) &= (\tilde{p}-p_h, w) \qquad \forall w \in L^2_0(\Omega),\ \text{a.e. } t\in(0,T], \\ (a^{-1}(\mathcal{C}_h(c_h))\xi, v) - (\psi, \nabla\cdot v) &= 0 \qquad \forall v \in H_0(\mathrm{div};\Omega),\ \text{a.e. } t\in(0,T]\end{aligned}$$

and then by setting $v = \tilde{u} - u_h$ and $w = \tilde{p} - p_h$ and subtracting the two equations

from each other we find

$$\begin{aligned}\|\tilde{p}-p_h\|^2_{L^2(\Omega)} &= (\tilde{p}-p_h,\tilde{p}-p_h)\\ &= -(a^{-1}(\mathcal{C}l_h(c_h))\xi,\tilde{u}-u_h)+(\psi,\nabla\cdot(\tilde{u}-u_h))+(\nabla\cdot\xi,\tilde{p}-p_h)\\ &= -(a^{-1}(\mathcal{C}l_h(c_h))\xi-a^{-1}(c_h)v_h,(\tilde{u}-u_h))\\ &\quad+(\nabla\cdot(\xi-v_h),\tilde{p}-p_h)+(\nabla\cdot(\tilde{u}-u_h),\psi-w_h)\end{aligned}$$

where we have used (5.2.12) and (5.2.13). We now choose $w_h=\psi_I$ satisfying (5.1.10) on each element and $v_h=\Pi_h^{\mathcal{RT}}(\xi)\in U$ by construction of the projection. Then using (5.2.2) with $v=\xi-\Pi_h^{\mathcal{RT}}(\xi)$ and (5.1.2) gives

$$\begin{aligned}&\|\tilde{p}-p_h\|^2_{L^2(\Omega)}\\ &= -(\nabla\cdot(\xi-\Pi_h^{\mathcal{RT}}(\xi)),p_h)-(\rho(c_h)g,\xi-\Pi_h^{\mathcal{RT}}(\xi))+(\nabla\cdot(\tilde{u}-u_h),\psi-\psi_I)\\ &\quad+(a^{-1}(c_h)u_h,\xi-\Pi_h^{\mathcal{RT}}(\xi))+((a^{-1}(c_h)-a^{-1}(\mathcal{C}l_h(c_h)))\xi,\tilde{u}-u_h)\\ &= (a^{-1}(c_h)u_h-\rho(c_h)g,\xi-\Pi_h^{\mathcal{RT}}(\xi))+((a^{-1}(c_h)-a^{-1}(\mathcal{C}l_h(c_h)))\xi,\tilde{u}-u_h)\\ &\quad+(\nabla\cdot(\tilde{u}-u_h),\psi-\psi_I)\end{aligned}$$

where we have also used (1.3.25). For the first term we use (5.1.3) and the regularity of ψ to show

$$\begin{aligned}&\sum_{E\in\mathcal{T}_h}\left(a^{-1}(c_h)u_h-\rho(c_h)g,\xi-\Pi_h^{\mathcal{RT}}(\xi)\right)_E\\ &\le\sum_{E\in\mathcal{T}_h}\|a^{-1}(c_h)u_h-\rho(c_h)g\|_{L^2(E)}\|\xi-\Pi_h^{\mathcal{RT}}(\xi)\|_{L^2(E)}\\ &\le C\left(\sum_{E\in\mathcal{T}_h}h_E^2\|a^{-1}(c_h)u_h-\rho(c_h)g\|^2_{L^2(E)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\tilde{p}-p_h\|^2_{L^2(E)}\right)^{1/2}.\end{aligned}$$

For the last term we use (5.1.10) and (5.2.1) to give

$$\begin{aligned}
&(\nabla\cdot(\tilde{u}-u_h),\psi-\psi_I)\\
&\leq C\sum_{E\in\mathcal{T}_h} h_E^2\|\psi\|_{H^2(E)}\|q^I-q^P-\nabla\cdot u_h\|_{L^2(E)}\\
&\leq C\left(\sum_{E\in\mathcal{T}_h} h_E^4\|q^I-q^P-\nabla\cdot u_h\|^2_{L^2(E)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\tilde{p}-p_h\|^2_{L^2(E)}\right)^{1/2}.
\end{aligned}$$

For the remaining term we have

$$\begin{aligned}
&((a^{-1}(c_h)-a^{-1}(\mathcal{C}l_h(c_h)))\xi,\tilde{u}-u_h)\\
&\leq\sum_{E\in\mathcal{T}_h}\|a^{-1}(c_h)-a^{-1}(\mathcal{C}l_h(c_h))\|_{L^\infty(E)}\|\xi\|_{L^2(E)}\|\tilde{u}-u_h\|_{L^2(E)}\\
&\leq C\|a^{-1}(c_h)-a^{-1}(\mathcal{C}l_h(c_h))\|_{L^\infty(\Omega)}\left(\sum_{E\in\mathcal{T}_h}\|\tilde{u}-u_h\|^2_{L^2(E)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\tilde{p}-p_h\|^2_{L^2(E)}\right)^{1/2}.
\end{aligned}$$

By combining terms and cancelling by $\|\tilde{p}-p_h\|$ we we reach

$$\sum_{E\in\mathcal{T}_h}\|\tilde{p}-p_h\|^2_{L^2(E)}\leq \mathtt{R_p}+\mathtt{R_a}\sum_{E\in\mathcal{T}_h}\|\tilde{u}-u_h\|^2_{L^2(E)} \tag{5.2.14}$$

where

$$\begin{aligned}
\mathtt{R_p}:=C\Bigg(&\sum_{E\in\mathcal{T}_h} h_E^2\|a^{-1}(c_h)u_h-\rho(c_h)g\|^2_{L^2(E)}\\
&+\sum_{E\in\mathcal{T}_h} h_E^4\|q^I-q^P-\nabla\cdot u_h\|^2_{L^2(E)}\Bigg)
\end{aligned} \tag{5.2.15}$$

and

$$\mathtt{R_a}:=C\|a^{-1}(c_h)-a^{-1}(\mathcal{C}l_h(c_h))\|^2_{L^\infty(\Omega)}. \tag{5.2.16}$$

We now seek to bound $\|\tilde{u} - u_h\|_{L^2(E)}$. We use (5.2.2) to show

$$\begin{aligned}
&(a^{-1}(c_h)(\tilde{u} - u_h), v) - (\nabla \cdot v, \tilde{p} - p_h) \\
&= -(a^{-1}(c_h)u_h, v) + (\nabla \cdot v, p_h) + (a^{-1}(c_h)\tilde{u}, v) - (\nabla \cdot v, \tilde{p}) \\
&= -(a^{-1}(c_h)u_h, v) + (\nabla \cdot v, p_h) + (\rho(c_h)g, v).
\end{aligned}$$

We employ again the Raviart-Thomas projection. By choosing $w = \tilde{p} - p_h$ and $v = \tilde{u} - u_h - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)$ we find

(5.2.17)

$$\begin{aligned}
&(a^{-1}(c_h)(\tilde{u} - u_h), (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) \\
&\quad -(\nabla \cdot ((\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)), \tilde{p} - p_h) = -(a^{-1}(c_h)u_h, (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) \\
&\qquad\qquad + (\nabla \cdot ((\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)), p_h) \\
&\qquad\qquad + (\rho(c_h)g, (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h))
\end{aligned}$$

and using (5.2.1)

(5.2.18)
$$\begin{aligned}
(\nabla \cdot (\tilde{u} - u_h), \tilde{p} - p_h) &= (\nabla \cdot \tilde{u}, \tilde{p} - p_h) - (\nabla \cdot u_h, \tilde{p} - p_h) \\
&= (q^I - q^P - \nabla \cdot u_h, \tilde{p} - p_h).
\end{aligned}$$

With the boundedness of a^{-1} and using in turn (5.2.17), (5.2.18), (5.2.13) and (5.1.2)

gives

$$
\begin{aligned}
&\alpha_\circ \|\tilde{u} - u_h\|^2_{L^2(\Omega)} \\
&\leq (a^{-1}(c_h)(\tilde{u} - u_h), \tilde{u} - u_h) \\
&= (a^{-1}(c_h)(\tilde{u} - u_h), (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) + (a^{-1}(c_h)(\tilde{u} - u_h), \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) \\
&= -(a^{-1}(c_h)u_h, (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) + (\nabla \cdot ((\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)), p_h) \\
&\quad + (\rho(c_h)g, (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) + (\nabla \cdot ((\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)), \tilde{p} - p_h) \\
&\quad + (a^{-1}(c_h)(\tilde{u} - u_h), \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) \\
&= -(a^{-1}(c_h)u_h, (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) + (q^I - q^P - \nabla \cdot u_h, \tilde{p} - p_h) \\
&\quad + (a^{-1}(c_h)(\tilde{u} - u_h), \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) - (\nabla \cdot (\Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)), \tilde{p} - p_h) \\
&\quad + (\nabla \cdot ((\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)), p_h) + (\rho(c_h)g, (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) \\
&= (q^I - q^p - \nabla \cdot u_h, \tilde{p} - p_h) \\
&\qquad - (a^{-1}(c_h)u_h - \rho(c_h)g, (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)).
\end{aligned}
$$

We bound the first of the terms using Young's inequality and (5.2.14) to give

$$
\begin{aligned}
&(q^I - q^p - \nabla \cdot u_h, \tilde{p} - p_h) \\
&\leq \frac{1}{2} \sum_{E \in \mathcal{T}_h} \left(\|q^I - q^P - \nabla \cdot u_h\|^2_{L^2(E)} + \|\tilde{p} - p_h\|^2_{L^2(E)} \right) \\
&\leq \frac{1}{2} \sum_{E \in \mathcal{T}_h} \left(\|q^I - q^P - \nabla \cdot u_h\|^2_{L^2(E)} + \mathtt{R_a} \|\tilde{u} - u_h\|^2_{L^2(E)} \right) + \frac{1}{2} \mathtt{R_p}.
\end{aligned}
$$

Using Young's inequality and (5.1.3) we have

$$
\begin{aligned}
&- (a^{-1}(c_h)u_h - \rho(c_h)g, (\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)) \\
&\leq \sum_{E \in \mathcal{T}_h} \frac{1}{4\varepsilon} \|a^{-1}(c_h)u_h - \rho(c_h)g\|^2_{L^2(E)} + \varepsilon \|(\tilde{u} - u_h) - \Pi_h^{\mathcal{RT}}(\tilde{u} - u_h)\|^2_{L^2(E)} \\
&\leq \sum_{E \in \mathcal{T}_h} \frac{1}{4\varepsilon} \|a^{-1}(c_h)u_h - \rho(c_h)g\|^2_{L^2(E)} + C\varepsilon h_E^2 \|\tilde{u} - u_h\|^2_{H^1(E)} \\
&\leq \sum_{E \in \mathcal{T}_h} \frac{1}{4\varepsilon} \|a^{-1}(c_h)u_h - \rho(c_h)g\|^2_{L^2(E)} + C\varepsilon h_E^2 \|q^I - q^P - \nabla \cdot u_h\|^2_{L^2(E)} \\
&\qquad + C\varepsilon h_E^2 \|\tilde{u} - u_h\|^2_{L^2(E)}
\end{aligned}
$$

where we have used $|\tilde{u} - u_h|^2_{H^1(E)} = \|\nabla \cdot (\tilde{u} - u_h)\|^2_{L^2(E)} = \|q^I - q^P - \nabla \cdot u_h\|^2_{L^2(E)}$.

By Assumption 5.2.7 we may choose ε such that $\alpha_\circ > C\varepsilon h_E^2 + \frac{1}{2}\mathtt{R_a}$. Then by combining the previous equations we find

$$\sum_{E\in\mathcal{T}_h} (\alpha_\circ - C\varepsilon h_E^2 - \frac{1}{2}\mathtt{R_a})\|\tilde{u} - u_h\|^2_{L^2(E)} \leq \mathtt{R_u} + \frac{1}{2}\mathtt{R_p} \tag{5.2.19}$$

where

$$\begin{aligned} \mathtt{R_u} = &\sum_{E\in\mathcal{T}_h} \left(C\varepsilon h_E^2 + \frac{1}{2}\right) \|q^I - q^p - \nabla \cdot u_h\|^2_{L^2(E)} \\ &+ \sum_{E\in\mathcal{T}_h} \frac{1}{4\varepsilon}\|a^{-1}(c_h)u_h - \rho(c_h)g\|^2_{L^2(E)}. \end{aligned} \tag{5.2.20}$$

We now use (5.2.19) to find a bound on $\|\tilde{p} - p_h\|_{L^2(E)}$ in (5.2.14), i.e,

$$\sum_{E\in\mathcal{T}_h} \|\tilde{p} - p_h\|^2_{L^2(E)} \leq \frac{\mathtt{R_a}}{\alpha_\circ - C\varepsilon h_E^2 - \frac{1}{2}\mathtt{R_a}}(\mathtt{R_u} + \frac{1}{2}\mathtt{R_p}) + \mathtt{R_p}.$$

We now combine (5.2.11), (5.2.14) and (5.2.19) to give

$$\begin{aligned} &\|\mathtt{E_u}\|^2_{L^2(\Omega)} + \|\mathtt{E_p}\|^2_{L^2(\Omega)} \\ &\leq \|u - \tilde{u}\|^2_{L^2(\Omega)} + \|\tilde{u} - u_h\|^2_{L^2(\Omega)} + \|p - \tilde{p}\|^2_{L^2(\Omega)} + \|\tilde{p} - p_h\|^2_{L^2(\Omega)} \\ &\leq \frac{C}{\alpha_\circ^2}\left(\rho^\circ\|g\|_{L^\infty(\Omega)} + \alpha^\circ\|u\|_{L^\infty(\Omega)}\right)^2 \|\mathtt{E_c}\|^2_{L^2(\Omega)} + C_u\mathtt{R_u} + C_p\mathtt{R_p} \end{aligned}$$

where C_u and C_p are the coefficients from (5.2.14) and (5.2.19). We show (5.2.9) by integrating over time and using the definition (1.3.5). □

5.3 An A Posteriori Estimator for the Concentration

We now present an a posteriori error estimator for the concentration. To do so we use the approach of, e.g., [126] and employ a "backward" parabolic equation. The coefficients of such an equation must be sufficiently regular to guarantee a bound of the type (5.3.5). We therefore employ the Clément interpolant, although other

interpolants or projections with appropriate approximation properties could be used.

Consider the following dual equation with bounded, continuous coefficients via the Clément interpolant:

$$\varphi\frac{\partial\zeta}{\partial t} + \nabla\cdot(\mathcal{C}l_h(u_h)\zeta) + \nabla\cdot(\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta) - q^I\zeta = \mathtt{E_c} \qquad \text{on } \Omega_T, \tag{5.3.1}$$

$$(\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta)\cdot n = 0 \qquad \text{on } \partial\Omega_T, \tag{5.3.2}$$

$$\zeta(T,x) = 0 \qquad \text{for } x\in\Omega. \tag{5.3.3}$$

We have the following theorem which extends that presented in Theorem 4.3.5 and can be found in [126, Theorem 4.1].

Theorem 5.3.4. *Given Assumption (A4) and the boundedness of q^I, the definition of $\mathbb{D}$ in (4.2.1), Ω a convex domain and $\mathtt{E_c}\in L^2((0,T];L^2(\Omega))$ there exists a unique solution ζ satisfying (5.3.1)-(5.3.3) with the regularity bounds*

$$\operatorname*{ess\,sup}_{t\in(0,T]}\|\zeta(t)\|_{H^1(\Omega)} + \|\zeta\|_{L^2((0,T];H^2(\Omega))} \le C\|\mathtt{E_c}\|_{L^2((0,T];L^2(\Omega))} \tag{5.3.5}$$

where C is a constant independent of $\mathtt{E_c}$.

Theorem 5.3.6. *Let the conditions of Assumption 5.1.11 hold and let (u_h,p_h,c_h) be the approximation defined in Definition 4.4.10. Then there exists constants C_4, C_5 and C_6 depending perhaps on the constants of the Clément approximation in Lemma 5.1.4, a trace inequality, the regularity bound (4.3.6) (restated in (5.3.5)), the polynomial degree of the approximation space and (5.1.9), but each independent of h, such that*

$$\|\mathtt{E_c}\|^2_{L^2((0,T];L^2(\Omega))} \le \mathscr{E}_c + C_4\sum_{E\in\mathcal{T}_h}\|\nabla c\|^2_{L^2((0,T];L^\infty(E))}\|\mathtt{E_u}\|^2_{L^2((0,T];L^2(E))} \tag{5.3.7}$$

where

$$
\begin{aligned}
\mathscr{E}_c := C_5 \sum_{E\in\mathcal{T}_h} \Big(& h_E^4 \|\mathtt{R_c}\|^2_{L^2((0,T];L^2(E))} \\
& + \|(\mathbb{D}(\mathcal{C}l_h(u_h)) - \mathbb{D}(u_h))\nabla c_h\|^2_{L^2((0,T];L^2(E))} \\
& + \|(\mathcal{C}l_h(u_h) - u_h)\cdot\nabla c_h\|^2_{L^2((0,T];L^2(E))} \\
& + \|\nabla c\|^2_{L^2((0,T];L^\infty(E))}\|\mathbb{D}(\mathcal{C}l_h(u_h)) - \mathbb{D}(u_h)\|^2_{L^2((0,T];L^2(E))} \\
& + \|\nabla c\|^2_{L^2((0,T];L^\infty(E))}\|\mathcal{C}l_h(u_h) - u_h\|^2_{L^2((0,T];L^2(E))}\Big) \\
+ C_6 \sum_{e\in\mathcal{E}_h} \Big(& h_e^3 \|[\![\mathbb{D}(u_h)\nabla c_h]\!]\|^2_{L^2((0,T];L^2(e))} \\
& + h_e\|[\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h)\}\!\}\|^2_{L^2((0,T];L^2(e))} \\
& + \|[\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h) - \mathbb{D}(\mathcal{C}l_h(u_h))\}\!\}\|^2_{L^2((0,T];L^2(e))} \\
& + h_e^3\|u_h\cdot[\![c_h]\!]\|^2_{L^2((0,T];L^2(e))} + \|[\![c_h]\!]\cdot(\mathcal{C}l_h(u_h) - u_h)\|^2_{L^2((0,T];L^2(e))}\Big)
\end{aligned}
\tag{5.3.8}
$$

and

$$
\mathtt{R_c} := \varphi\frac{\partial c_h}{\partial t} + u_h\cdot\nabla c_h - \nabla\cdot(\mathbb{D}(u_h)\nabla c_h) + q^I c_h - \hat{c}q^I. \tag{5.3.9}
$$

Proof. Using (5.3.1) we see

$$
\begin{aligned}
\|\mathtt{E_c}\|^2_{L^2(\Omega)} &= (\mathtt{E_c},\mathtt{E_c}) \\
&= \left(\varphi\frac{\partial\zeta}{\partial t} + \nabla\cdot(\mathcal{C}l_h(u_h)\zeta) + \nabla\cdot(\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta) - q^I\zeta, \mathtt{E_c}\right) \\
&= -\left(\varphi\frac{\partial\mathtt{E_c}}{\partial t},\zeta\right) - \frac{\mathrm{d}}{\mathrm{d}t}(\varphi\mathtt{E_c},\zeta) + (\mathtt{E_c},\nabla\cdot(\mathcal{C}l_h(u_h)\zeta)) - (q^I\zeta,\mathtt{E_c}) \\
&\quad - \sum_{E\in\mathcal{T}_h}\left[(\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\mathtt{E_c},\nabla\zeta)_E + \int_{\partial E}(\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta\mathtt{E_c})\cdot n\,\mathrm{d}s\right]
\end{aligned}
$$

where we have integrated two terms. By chosing $d = \mathcal{C}l_h(\zeta)$ in (4.4.2) and $d_h = \mathcal{C}l_h(\zeta)$ in (4.4.11), and subtracting the two equations we find

$$
\begin{aligned}
0 = & \left(\varphi\left(\frac{\partial c}{\partial t} - \frac{\partial c_h}{\partial t}\right), \mathcal{C}l_h(\zeta)\right) + \mathcal{B}_d(c,\mathcal{C}l_h(\zeta);u) - \mathcal{B}_d(c_h,\mathcal{C}l_h(\zeta);u_h) \\
& + \mathcal{B}_{cq}^{\mathrm{alt}}(c,\mathcal{C}l_h(\zeta);u) - \mathcal{B}_{cq}^{\mathrm{alt}}(c_h,\mathcal{C}l_h(\zeta);u_h)
\end{aligned}
\tag{5.3.10}
$$

where we have used the fact that the bilinear form for the approximate problem reduces to the weak form (4.4.2) when using continuous arguments. We add this to the above result giving

(5.3.11)

$$
\begin{aligned}
&\|\mathtt{E_c}\|^2_{L^2(\Omega)} \\
&= -\left(\varphi\frac{\partial \mathtt{E_c}}{\partial t}, \zeta - \mathcal{C}l_h(\zeta)\right) - \frac{\mathrm{d}}{\mathrm{d}t}\left(\varphi \mathtt{E_c}, \zeta\right) - \left(q^I \mathtt{E_c}, \zeta - \mathcal{C}l_h(\zeta)\right) \\
&\quad + \sum_{E\in\mathcal{T}_h} \left(\mathbb{D}(u)\nabla c, \nabla \mathcal{C}l_h(\zeta)\right)_E - \left(\mathbb{D}(u_h)\nabla c_h, \nabla \mathcal{C}l_h(\zeta)\right)_E - \left(\mathbb{D}(\mathcal{C}l_h(u_h))\nabla \mathtt{E_c}, \nabla\zeta\right)_E \\
&\quad + \left(\mathtt{E_c}, \nabla\cdot\left(\mathcal{C}l_h(u_h)\zeta\right)\right) + \sum_{E\in\mathcal{T}_h}\left(u\cdot\nabla c - u_h\cdot\nabla c_h, \mathcal{C}l_h(\zeta)\right)_E \\
&\quad + \sum_{e\in\mathcal{E}_h^o}\int_e [\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h)\nabla\mathcal{C}l_h(\zeta)\}\!\}\,\mathrm{d}s + \sum_{e\in\mathcal{E}_h^o}\int_e u_h\cdot[\![c_h]\!]\mathcal{C}l_h(\zeta)\,\mathrm{d}s \\
&\quad + \sum_{E\in\mathcal{T}_h}\int_{\partial E}\left(\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta\mathtt{E_c}\right)\cdot n\,\mathrm{d}s
\end{aligned}
$$

where we have used the regularity of ζ and $\mathcal{C}l_h(\zeta)$ to simplify terms. We examine each of these terms in turn. Using (1.3.25) we have

$$
\begin{aligned}
&\sum_{E\in\mathcal{T}_h}\int_{\partial E}\left(\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta\mathtt{E_c}\right)\cdot n\,\mathrm{d}s \\
&= \sum_{e\in\mathcal{E}_h}\int_e [\![\mathtt{E_c}]\!]\cdot\{\!\{\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta\}\!\}\,\mathrm{d}s + \sum_{e\in\mathcal{E}_h^o}\int_e \{\!\{\mathtt{E_c}\}\!\}[\![\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta]\!]\,\mathrm{d}s \\
&= -\sum_{e\in\mathcal{E}_h}\int_e [\![c_h]\!]\cdot\{\!\{\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta\}\!\}\,\mathrm{d}s
\end{aligned}
\tag{5.3.12}
$$

where the negative sign comes from the definition of $[\![\cdot]\!]$. Combining this with the similar term in (5.3.11) gives

$$
\begin{aligned}
&\sum_{e\in\mathcal{E}_h^o}\int_e [\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h)\nabla\mathcal{C}l_h(\zeta)\}\!\} - [\![c_h]\!]\cdot\{\!\{\mathbb{D}(\mathcal{C}l_h(u_h))\nabla\zeta\}\!\}\,\mathrm{d}s \\
&= \sum_{e\in\mathcal{E}_h^o}\int_e [\![c_h]\!]\cdot\{\!\{(\mathbb{D}(u_h)-\mathbb{D}(\mathcal{C}l_h(u_h)))\nabla\zeta\}\!\} - [\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h)\nabla(\zeta-\mathcal{C}l_h(\zeta))\}\!\}\,\mathrm{d}s.
\end{aligned}
$$

For the next term in (5.3.11) we integrate and use (1.3.25) to find

$$(\mathtt{E_c}, \nabla\cdot(\mathcal{C}\!l_h(u_h)\zeta)) = -\sum_{E\in\mathcal{T}_h}(\mathcal{C}\!l_h(u_h)\cdot\nabla\mathtt{E_c},\zeta)_E - \sum_{e\in\mathcal{E}_h}\int_e \mathcal{C}\!l_h(u_h)\cdot[\![c_h]\!]\zeta\,\mathrm{d}s. \tag{5.3.13}$$

Combining the first part with another of the terms from (5.3.11) gives

$$\begin{aligned}
&\sum_{E\in\mathcal{T}_h}(u\cdot\nabla c - u_h\cdot\nabla c_h, \mathcal{C}\!l_h(\zeta))_E - (\mathcal{C}\!l_h(u_h)\cdot\nabla\mathtt{E_c},\zeta)_E\\
&= \sum_{E\in\mathcal{T}_h}\Big[(-u\cdot\nabla c, \zeta-\mathcal{C}\!l_h(\zeta))_E + ((u-\mathcal{C}\!l_h(u_h))\cdot\nabla c,\zeta)_E\\
&\qquad + (u_h\cdot\nabla c_h, \zeta-\mathcal{C}\!l_h(\zeta))_E + ((\mathcal{C}\!l_h(u_h)-u_h)\cdot\nabla c_h,\zeta)_E\Big]
\end{aligned}$$

and as $u_h\cdot n$ is continuous through the definition of the RT space we combine the second part of (5.3.13) with the similar term in (5.3.11) giving

$$\begin{aligned}
&\sum_{e\in\mathcal{E}_h^o}\int_e u_h\cdot[\![c_h]\!]\mathcal{C}\!l_h(\zeta) - \mathcal{C}\!l_h(u_h)\cdot[\![c_h]\!]\zeta\,\mathrm{d}s\\
&= -\sum_{e\in\mathcal{E}_h^o}\int_e [\![c_h]\!]\cdot((\zeta-\mathcal{C}\!l_h(\zeta))u_h + (\mathcal{C}\!l_h(u_h)-u_h)\zeta)\,\mathrm{d}s.
\end{aligned}$$

We rewrite the following term:

$$\begin{aligned}
&\sum_{E\in\mathcal{T}_h}(\mathbb{D}(u)\nabla c, \nabla\mathcal{C}\!l_h(\zeta))_E - (\mathbb{D}(u_h)\nabla c_h, \nabla\mathcal{C}\!l_h(\zeta))_E - (\mathbb{D}(\mathcal{C}\!l_h(u_h))\nabla\mathtt{E_c},\nabla\zeta)_E\\
&= \sum_{E\in\mathcal{T}_h}((\mathbb{D}(u)-\mathbb{D}(\mathcal{C}\!l_h(u_h)))\nabla c,\nabla\zeta)_E + (\mathbb{D}(u_h)\nabla c_h - \mathbb{D}(u)\nabla c, \nabla(\zeta-\mathcal{C}\!l_h(\zeta)))_E\\
&\qquad + ((\mathbb{D}(\mathcal{C}\!l_h(u_h))-\mathbb{D}(u_h))\nabla c_h,\nabla\zeta)_E\,.
\end{aligned}$$

Now integrating the second inner product and using (1.3.25) we find

$$\begin{aligned}
&\sum_{E\in\mathcal{T}_h}(\mathbb{D}(u_h)\nabla c_h - \mathbb{D}(u)\nabla c, \nabla(\zeta-\mathcal{C}\!l_h(\zeta)))_E\\
&= \sum_{E\in\mathcal{T}_h}(\nabla\cdot(\mathbb{D}(u)\nabla c - \mathbb{D}(u_h)\nabla c_h), \zeta-\mathcal{C}\!l_h(\zeta))_E + \sum_{e\in\mathcal{E}_h^o}\int_e [\![\mathbb{D}(u_h)\nabla c_h]\!](\zeta-\mathcal{C}\!l_h(\zeta))\,\mathrm{d}s.
\end{aligned}$$

Using the definition (5.3.9) and the original transport equation (1.1.3) we have

shown that (5.3.11) can be written as

(5.3.14)

$$
\begin{aligned}
&\|\mathtt{E_c}\|^2_{L^2(\Omega)} \\
&= (\mathtt{R_c}, \zeta - \mathcal{C}l_h(\zeta)) - \frac{\mathrm{d}}{\mathrm{d}t}(\varphi \mathtt{E_c}, \zeta) + \sum_{E\in\mathcal{T}_h} \Big[((\mathbb{D}(u) - \mathbb{D}(\mathcal{C}l_h(u_h)))\nabla c, \nabla\zeta)_E \\
&\quad + ((\mathbb{D}(\mathcal{C}l_h(u_h)) - \mathbb{D}(u_h))\nabla c_h, \nabla\zeta)_E + ((u - \mathcal{C}l_h(u_h))\cdot\nabla c, \zeta)_E \\
&\quad + ((\mathcal{C}l_h(u_h) - u_h)\cdot\nabla c_h, \zeta)_E \Big] + \sum_{e\in\mathcal{E}_h^o} \int_e [\![\mathbb{D}(u_h)\nabla c_h]\!](\zeta - \mathcal{C}l_h(\zeta))\,\mathrm{d}s \\
&\quad + \sum_{e\in\mathcal{E}_h^o} \int_e [\![c_h]\!]\cdot \{\!\{(\mathbb{D}(u_h) - \mathbb{D}(\mathcal{C}l_h(u_h)))\nabla\zeta\}\!\} - [\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h)\nabla(\zeta - \mathcal{C}l_h(\zeta))\}\!\}\,\mathrm{d}s \\
&\quad - \sum_{e\in\mathcal{E}_h^o} \int_e [\![c_h]\!]\cdot((\zeta - \mathcal{C}l_h(\zeta))u_h + (\mathcal{C}l_h(u_h) - u_h)\zeta)\,\mathrm{d}s.
\end{aligned}
$$

We examine each of these terms using the Cauchy-Schwarz inequality, interpolation approximations (5.1.5) and (5.1.6) and regularity bound (5.3.5). This gives

$$
\begin{aligned}
(\mathtt{R_c}, \zeta - \mathcal{C}l_h(\zeta)) &\le C \sum_{E\in\mathcal{T}_h} \|\mathtt{R_c}\|_{L^2(E)} \|\zeta - \mathcal{C}l_h(\zeta)\|_{L^2(E)} \\
&\le C \sum_{E\in\mathcal{T}_h} h_E^2 \|\mathtt{R_c}\|_{L^2(E)} \|\zeta\|_{H^2(\Delta_E)} \\
&\le C \left(\sum_{E\in\mathcal{T}_h} h_E^4 \|\mathtt{R_c}\|^2_{L^2(E)} \right)^{1/2} \left(\sum_{E\in\mathcal{T}_h} \|\mathtt{E_c}\|^2_{L^2(E)} \right)^{1/2},
\end{aligned}
$$

$$
\begin{aligned}
&\sum_{E\in\mathcal{T}_h} ((\mathbb{D}(\mathcal{C}l_h(u_h)) - \mathbb{D}(u_h))\nabla c_h, \nabla\zeta)_E \\
&\le \left(\sum_{E\in\mathcal{T}_h} \|(\mathbb{D}(\mathcal{C}l_h(u_h)) - \mathbb{D}(u_h))\nabla c_h\|^2_{L^2(E)} \right)^{1/2} \left(\sum_{E\in\mathcal{T}_h} \|\mathtt{E_c}\|^2_{L^2(E)} \right)^{1/2}
\end{aligned}
$$

and

$$\sum_{E\in\mathcal{T}_h} \left((\mathcal{C}l_h(u_h) - u_h)\cdot\nabla c_h, \zeta\right)_E$$
$$\leq \left(\sum_{E\in\mathcal{T}_h} \|(\mathcal{C}l_h(u_h) - u_h)\cdot\nabla c_h\|^2_{L^2(E)}\right)^{1/2} \left(\sum_{E\in\mathcal{T}_h} \|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}.$$

The following cell terms are trickier. We add and remove terms then use the approximation properties of the Clément interpolant and Lemma 5.1.8 to show

$$\sum_{E\in\mathcal{T}_h} \left((\mathbb{D}(u) - \mathbb{D}(\mathcal{C}l_h(u_h)))\nabla c, \nabla\zeta\right)_E$$
$$= \sum_{E\in\mathcal{T}_h} \left((\mathbb{D}(u) - \mathbb{D}(u_h) + \mathbb{D}(u_h) - \mathbb{D}(\mathcal{C}l_h(u_h)))\nabla c, \nabla\zeta\right)_E$$
$$\leq \sum_{E\in\mathcal{T}_h} \|\mathbb{D}(u) - \mathbb{D}(u_h)\|_{L^2(E)}\|\nabla c\|_{L^\infty(E)}\|\nabla\zeta\|_{L^2(E)}$$
$$+ \sum_{E\in\mathcal{T}_h} \|\mathbb{D}(u_h) - \mathbb{D}(\mathcal{C}l_h(u_h))\|_{L^2(E)}\|\nabla c\|_{L^\infty(E)}\|\nabla\zeta\|_{L^2(E)}$$
$$\leq C\left(\sum_{E\in\mathcal{T}_h} \|\nabla c\|^2_{L^\infty(E)}\|\mathbb{D}(u_h) - \mathbb{D}(\mathcal{C}l_h(u_h))\|^2_{L^2(E)}\right)^{1/2} \left(\sum_{E\in\mathcal{T}_h} \|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}$$
$$+ C\left(\sum_{E\in\mathcal{T}_h} \|\nabla c\|^2_{L^\infty(E)}\right)^{1/2} \left(\sum_{E\in\mathcal{T}_h} \|\mathtt{E_u}\|^2_{L^2(E)}\right)^{1/2} \left(\sum_{E\in\mathcal{T}_h} \|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}$$

and

$$
\begin{aligned}
&\sum_{E\in\mathcal{T}_h}((u-\mathcal{C}l_h(u_h))\cdot\nabla c,\zeta)_E\\
&=\sum_{E\in\mathcal{T}_h}((u-u_h+u_h-\mathcal{C}l_h(u_h))\cdot\nabla c,\zeta)_E\\
&\le\sum_{E\in\mathcal{T}_h}\|u-u_h\|_{L^2(E)}\|\nabla c\|_{L^\infty(E)}\|\zeta\|_{L^2(E)}\\
&\qquad+\sum_{E\in\mathcal{T}_h}\|u_h-\mathcal{C}l_h(u_h)\|_{L^2(E)}\|\nabla c\|_{L^\infty(E)}\|\zeta\|_{L^2(E)}\\
&\le C\left(\sum_{E\in\mathcal{T}_h}\|\nabla c\|^2_{L^\infty(E)}\|u_h-\mathcal{C}l_h(u_h)\|^2_{L^2(E)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}\\
&\qquad+C\left(\sum_{E\in\mathcal{T}_h}\|\nabla c\|^2_{L^\infty(E)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\mathtt{E_u}\|^2_{L^2(E)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}.
\end{aligned}
$$

Now we examine the terms on the edges. Firstly using (5.1.7) and the shape regularity of the mesh we have

$$
\begin{aligned}
&\sum_{e\in\mathcal{E}_h^o}\int_e[\![\mathbb{D}(u_h)\nabla c_h]\!](\zeta-\mathcal{C}l_h(\zeta))\,\mathrm{d}s\\
&\le\sum_{e\in\mathcal{E}_h^o}\|[\![\mathbb{D}(u_h)\nabla c_h]\!]\|_{L^2(e)}\|\zeta-\mathcal{C}l_h(\zeta)\|_{L^2(e)}\\
&\le C\sum_{e\in\mathcal{E}_h^o}\|[\![\mathbb{D}(u_h)\nabla c_h]\!]\|_{L^2(e)}\sum_{E\in\mathcal{T}_h}h_E^{3/2}|\zeta|_{H^2(\Delta_e)}\\
&\le C\left(\sum_{e\in\mathcal{E}_h^o}h_e^3\|[\![\mathbb{D}(u_h)\nabla c_h]\!]\|^2_{L^2(e)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2},
\end{aligned}
$$

$$
\begin{aligned}
&-\sum_{e\in\mathcal{E}_h}\int_e[\![c_h]\!]\cdot\{\!\{(\mathbb{D}(u_h)\nabla(\zeta-\mathcal{C}l_h(\zeta))\}\!\}\,\mathrm{d}s\\
&\le C\left(\sum_{e\in\mathcal{E}_h}h_e\|[\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h)\}\!\}\|^2_{L^2(e)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}
\end{aligned}
$$

and

$$
\begin{aligned}
&-\sum_{e\in\mathcal{E}_h}\int_e u_h\cdot[\![c_h]\!](\zeta-\mathcal{C}l_h(\zeta))\,\mathrm{d}s\\
&\leq C\left(\sum_{e\in\mathcal{E}_h}h_e^3\|u_h\cdot[\![c_h]\!]\|^2_{L^2(e)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}.
\end{aligned}
$$

For the remaining terms we use a trace inequality on each element to show

$$
\begin{aligned}
&\sum_{e\in\mathcal{E}_h}\int_e[\![c_h]\!]\cdot(\mathcal{C}l_h(u_h)-u_h)\zeta\,\mathrm{d}s\\
&\leq\sum_{e\in\mathcal{E}_h}\|[\![c_h]\!]\cdot(\mathcal{C}l_h(u_h)-u_h)\|_{L^2(e)}\|\zeta\|_{L^2(e)}\\
&\leq C\sum_{e\in\mathcal{E}_h}\|[\![c_h]\!]\cdot(\mathcal{C}l_h(u_h)-u_h)\|_{L^2(e)}\sum_{E\in\mathcal{T}_h}\|\zeta\|^{1/2}_{L^2(E)}\|\zeta\|^{1/2}_{H^1(E)}\\
&\leq C\left(\sum_{e\in\mathcal{E}_h}\|[\![c_h]\!]\cdot(\mathcal{C}l_h(u_h)-u_h)\|^2_{L^2(e)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}
\end{aligned}
$$

and similarly

$$
\begin{aligned}
&\sum_{e\in\mathcal{E}_h}\int_e[\![c_h]\!]\cdot\{\!\{(\mathbb{D}(u_h)-\mathbb{D}(\mathcal{C}l_h(u_h)))\nabla\zeta\}\!\}\,\mathrm{d}s\\
&\leq C\left(\sum_{e\in\mathcal{E}_h}\|[\![c_h]\!]\cdot\{\!\{\mathbb{D}(u_h)-\mathbb{D}(\mathcal{C}l_h(u_h))\}\!\}\|^2_{L^2(e)}\right)^{1/2}\left(\sum_{E\in\mathcal{T}_h}\|\mathtt{E_c}\|^2_{L^2(E)}\right)^{1/2}.
\end{aligned}
$$

We now integrate over time. The remaining term is zero via Assumption (A6), i.e., $c_0\in V_{\mathrm{dG}}$ so

$$
-\frac{\mathrm{d}}{\mathrm{d}t}(\varphi\mathtt{E_c},\zeta)=(\varphi(c_0-c_{0,h}),\zeta(0)-\mathcal{C}l_h(\zeta(0)))=0.
$$

We find (5.3.7) by combining each of the terms. □

5.4 An A Posteriori Estimator for the Coupled Problem

We now combine the results of Theorems 5.2.8 and 5.3.6. We first make an assumption about the size of the terms in those theorems.

Assumption 5.4.1. *We assume that we can find constants Λ_{up} and Λ_c such that*

$$\frac{C_1}{\alpha_\circ^2}\left(\rho^\circ\|g\|_{L^\infty(\Omega)}+\alpha^\circ\|u\|_{L^\infty(\Omega)}\right)^2 \leq \frac{\Lambda_{up}}{1+\Lambda_c},$$
$$C_4\sum_{E\in\mathcal{T}_h}\|\nabla c\|^2_{L^2((0,T];L^\infty(E))} \leq \frac{\Lambda_c}{1+\Lambda_{up}}$$

where C_1 and C_4 are defined in Theorems 5.2.8 and 5.3.6 respectively.

We will consider conditions to achieve assumptions of this type more carefully in Chapter 6, Lemma 6.1.6. For now we remark that we have control over the choice of ε in (5.2.19) which may help us satisfy the assumption.

Theorem 5.4.2. *Let the conditions of Assumptions 5.1.11 and 5.4.1 hold, and let (u_h, p_h, c_h) be the approximation defined in Definition 4.4.10. In the notation of Theorems 5.2.8 and 5.3.6 we have*

$$\begin{aligned}&\|\mathtt{E_c}\|^2_{L^2((0,T];L^2(\Omega))}+\|\mathtt{E_u}\|^2_{L^2((0,T];L^2(\Omega))}+(1+\Lambda_c)\|\mathtt{E_p}\|^2_{L^2((0,T];L^2(\Omega))}\\&\leq(1+\Lambda_{up})\mathscr{E}_c+(1+\Lambda_c)\mathscr{E}_{up}.\end{aligned}\tag{5.4.3}$$

Proof. Multiply (5.3.7) by $(1+\Lambda_{up})$ and (5.2.9) by $(1+\Lambda_c)$. Adding the resulting terms and rearranging yields (5.4.3). □

Chapter 6

A Posteriori Error Estimators for the Incompressible Miscible Displacement Problem in Weighted Spaces

The regularity guaranteed (or assumed) in Chapter 5 may not hold in less regular domains (such as those with re-entrant corners) or where the coefficients are discontinuous or singular. In order to discuss such cases we consider stationary coupled problems in this chapter, firstly in an abstract setting and then for a simple example. This example motivates the use of weighted spaces, which are introduced and described. We discuss aspects of weighted spaces which may be useful in the study of a posteriori error estimators.

The work in this Chapter has been published in part in [49].

6.1 An Abstract Discussion

Consider the real vector spaces $C \subset \mathbf{C}$ and C^+ as well as $P \subset \mathbf{P}$ and P^+. Also consider the operators $\Phi : P \times \mathbf{C} \to P^+$ and $\Psi : C \times \mathbf{P} \to C^+$. Then the *continuous*

problem is to find a solution $(c, p) \in C \times P$ of the system of equations

$$\Phi(p; c) = f, \qquad \Psi(c; p) = g \tag{6.1.1}$$

for some $f \in P^+$, $g \in C^+$. We assume that such a solution exists.

Suppose that $C_h \subset \mathbf{C}$ and $P_h \subset \mathbf{P}$ are finite dimensional spaces indexed by h. Consider the operators $\Phi_h : P_h \times C_h \to P^+$ and $\Psi_h : C_h \times P_h \to C^+$. The *discrete problem* is to find a solution $(c_h, p_h) \in C_h \times P_h$, solving

$$\Phi_h(p_h; c_h) = f_h, \qquad \Psi_h(c_h; p_h) = g_h. \tag{6.1.2}$$

It is assumed that discrete solutions exist.

The *auxiliary problem* is to find the solution $(\tilde{c}, \tilde{p}) \in C \times P$ of the system of equations

$$\Phi(\tilde{p}; c_h) = f, \qquad \Psi(\tilde{c}; p_h) = g. \tag{6.1.3}$$

If (6.1.2) does not define c_h and p_h uniquely, then $\tilde{p} = \tilde{p}(c_h)$ and $\tilde{c} = \tilde{c}(p_h)$ may depend on the choice of c_h and p_h. It is, however, assumed that for given c_h and p_h there exist unique $\tilde{c}$ and $\tilde{p}$. The discrete spaces have mesh-dependent norms $\|\cdot\|_{C_h}$ and $\|\cdot\|_{P_h}$ which have extensions to $C + C_h$ and $P + P_h$.

Assumption 6.1.4 (Coupling Assumption)**.** *Let $(c, p) \in C \times P$ be a solution of (6.1.1). We assume that there exists $\gamma_c, \gamma_p \in \mathbb{R}$ such that $\gamma_c \gamma_p < 1$ and*

$$\begin{aligned} \Phi(\tilde{w}; d_h) = f \Rightarrow \|p - \tilde{w}\|_{P_h}^2 \le \gamma_p \|c - d_h\|_{C_h}^2, \qquad & \forall \tilde{w} \in P,\ \forall d_h \in C + C_h, \\ \Psi(\tilde{d}; w_h) = g \Rightarrow \|c - \tilde{d}\|_{C_h}^2 \le \gamma_c \|p - w_h\|_{P_h}^2, \qquad & \forall \tilde{d} \in C,\ \forall w_h \in P + P_h. \end{aligned}$$

With the coupling assumption there is only one exact solution: Suppose there are two solutions (c, p) and (c°, p°) of (6.1.1). Then with Assumption 6.1.4 we have

$$\|p - p^\circ\|_{P_h}^2 \le \gamma_p \|c - c^\circ\|_{C_h}^2 \le \gamma_p \gamma_c \|p - p^\circ\|_{P_h}^2.$$

As $\gamma_p\gamma_c < 1$ this can only be satisfied if (c, p) and (c°, p°) coincide.

Assumption 6.1.5. *Let (c_h, p_h) be a solution of (6.1.2) and $(\tilde{c}, \tilde{p})$ be a solution of (6.1.3). Then we assume that there exist a posteriori error estimators $\mathscr{E}_p(c_h, p_h)$ and $\mathscr{E}_c(c_h, p_h)$ such that*

$$\|p_h - \tilde{p}\|^2_{P_h} \leq \mathscr{E}_p(c_h, p_h),$$
$$\|c_h - \tilde{c}\|^2_{C_h} \leq \mathscr{E}_c(c_h, p_h).$$

Lemma 6.1.6. *There are positive constants Λ_p and Λ_c such that*

$$\gamma_p \leq \frac{\Lambda_p}{1+\Lambda_c} \qquad \text{and} \qquad \gamma_c \leq \frac{\Lambda_c}{1+\Lambda_p} \tag{6.1.7}$$

if and only if $\gamma_p\gamma_c < 1$.

Proof. The bounds (6.1.7) are satisfied as equalities if

$$\Lambda_p = \frac{(1+\gamma_p)\gamma_c}{1-\gamma_p\gamma_c}, \qquad \Lambda_c = \frac{(1+\gamma_c)\gamma_p}{1-\gamma_c\gamma_p}.$$

If $\gamma_p\gamma_c < 1$ then these Λ_p and Λ_c are positive. On the other hand, if (6.1.7) holds then

$$\gamma_p\gamma_c \leq \frac{\Lambda_p}{1+\Lambda_c}\frac{\Lambda_c}{1+\Lambda_p} = \frac{\Lambda_p\Lambda_c}{1+\Lambda_p+\Lambda_c+\Lambda_p\Lambda_c} < 1.$$

□

Note the similarity of this result to Assumption 5.4.1.

We show that Assumptions 6.1.4 and 6.1.5 lead to an a posteriori error indicator for the approximation of the solution of coupled system (6.1.1) using the discrete problem.

Theorem 6.1.8. *Let (c, p), (c_h, p_h) and $(\tilde{c}, \tilde{p})$ be as defined in (6.1.1)-(6.1.3) and let Assumptions 6.1.4 and 6.1.5 hold. Then we have the following a posteriori error bound:*

$$\|p - p_h\|^2_{P_h} + \|c - c_h\|^2_{C_h} \leq (1+\Lambda_c)\mathscr{E}_p(c_h, p_h) + (1+\Lambda_p)\mathscr{E}_c(c_h, p_h) \tag{6.1.9}$$

where Λ_c *and* Λ_p *are defined by* (6.1.7).

Proof. We have with the triangle inequality and the assumptions that

$$\|p - p_h\|_{P_h}^2 \leq \|p - \tilde{p}\|_{P_h}^2 + \|p_h - \tilde{p}\|_{P_h}^2 \leq \gamma_p \|c - c_h\|_{C_h}^2 + \mathscr{E}_p(c_h, p_h) \tag{6.1.10}$$

and similarly

$$\|c - c_h\|_{C_h}^2 \leq \gamma_c \|p - p_h\|_{P_h}^2 + \mathscr{E}_c(c_h, p_h). \tag{6.1.11}$$

Then with these results and (6.1.7) we have

$$\begin{aligned}
&(1 + \Lambda_c)\|p - p_h\|_{P_h}^2 + (1 + \Lambda_p)\|c - c_h\|_{C_h}^2 \\
&\leq \Lambda_p \|c - c_h\|_{C_h}^2 + (1 + \Lambda_c)\mathscr{E}_p(c_h, p_h) + \Lambda_c \|p - p_h\|_{P_h}^2 + (1 + \Lambda_p)\mathscr{E}_c(c_h, p_h)
\end{aligned}$$

and by rearrangement we have (6.1.9). □

6.2 The Stationary Incompressible Miscible Displacement Problem

In later sections we will discuss weighted spaces. In order to motivate that discussion we introduce the stationary incompressible miscible displacement problem in two dimensions and discuss a posteriori error estimators for the cG-cG problem (that is where pressure and concentration are both approximated in V_{cG}) in the abstract framework of the previous section.

Define the following operators on a bounded C^2-regular or convex polygonal domain $\Omega \in \mathbb{R}^2$:

$$\Phi(p; c) := -\nabla \cdot (a(c)\nabla p) = q^I - q^P, \tag{6.2.1}$$

$$\Psi(c; p) := -\nabla \cdot (\mathbb{D}(u)\nabla c) + \frac{1}{2} u \cdot \nabla c + \frac{1}{2}\nabla \cdot (uc) + \frac{1}{2}(q^I + q^P)c = \hat{c} q^I, \tag{6.2.2}$$

$$u = -a(c)\nabla p \tag{6.2.3}$$

subject to boundary conditions on $\partial\Omega$

$$u \cdot n = 0, \tag{6.2.4}$$

$$(\mathbb{D}(u)\nabla c) \cdot n = 0. \tag{6.2.5}$$

We make the same assumptions on the coefficients as in Chapter 4. Note that the analysis that follows may be extended quite simply to three dimensions. We focus on the two dimensional case to accommodate the weighted spaces in subsequent sections.

In the notation of Section 6.1 for the miscible displacement problem we identify p as the pressure and c as the concentration. Therefore Φ describes Darcy's law and incompressibility and Ψ describes the concentration equation. In such a context C_h and P_h are finite element approximation spaces, C and P are spaces leading to a natural notion of regularity for Ψ and Φ respectively, and $\mathbf{C}$ and $\mathbf{P}$ are spaces containing all C_h and P_h. Finally C^+ and P^+ are the co-domain of Ψ and Φ respectively. More specifically we have

$$C = W^{1,\infty}(\Omega), \quad \mathbf{C} = H^1(\Omega), \quad C^+ = L^2(\Omega) \quad \text{and} \quad C_h = V_{\mathrm{cG}}$$

for concentration and

$$P = W_0(\Omega), \quad \mathbf{P} = H^1(\Omega), \quad P^+ = L^2(\Omega) \quad \text{and } P_h = V_{\mathrm{cG}} \cap W_0(\Omega)$$

where

$$W_0(\Omega) := \left\{ w \in W^{1,\infty}(\Omega) : \int_\Omega w \, \mathrm{d}\boldsymbol{x} = 0 \right\}.$$

We norm the approximation spaces with $\|\cdot\|_{H^1(\Omega)}$.

For all $w \in W_0(\Omega)$ and $d \in W^{1,\infty}(\Omega)$ the weak forms of (6.2.1) and (6.2.2) are

given by

$$\Phi(p;c)[w] = (w \mapsto (a(c)\nabla p, \nabla w)), \tag{6.2.6}$$

$$\Psi(c;p)[d] = (d \mapsto (\mathbb{D}(u)\nabla c, \nabla d) + \frac{1}{2}(u \cdot \nabla c, d) - \frac{1}{2}(uc, \nabla d) + \frac{1}{2}((q^I + q^P)c, d)). \tag{6.2.7}$$

We assume that the exact solution (c, p) to (6.2.1)-(6.2.2) belongs to $W^{1,\infty}(\Omega) \times W^{1,\infty}(\Omega)$ (the well posedness of the model in $H^2(\Omega) \times H^2(\Omega)$ is established [105]). With the given boundary regularity (or convexity) this is not an unreasonable assumption. Following the analysis in, e.g., [102, Chapter 7] we have that the solution to each of (6.2.1) and (6.2.2) when uncoupled (with appropriate boundary conditions) has bounded derivatives. As previously remarked this does not hold for elliptic equations on non-convex domains and is a major motivator for development of an alternative approach.

We then define the following problems for stationary IMD.

Definition 6.2.8. *Define the continuous problem as: Find* $(c, p) \in W^{1,\infty}(\Omega) \times W_0$ *such that*

$$\begin{aligned} \Phi(p;c) &= q^I - q^P, \\ \Psi(c;p) &= \hat{c}q^I. \end{aligned} \tag{6.2.9}$$

Given the regularity of the discrete solution the weak operator can be used to define the discrete problem.

Definition 6.2.10. *Define the discrete problem: Find* $(c_h, p_h) \in V_{cG} \times (V_{cG} \cap W_0(\Omega))$ *such that*

$$\begin{aligned} \Phi(p_h;c_h)[w_h] &= q_h^I - q_h^P \qquad \forall w_h \in V_{cG}(\Omega) \cap W_0(\Omega), \\ \Psi(c_h;p_h)[d_h] &= \hat{c}_h q_h^I \qquad \forall d_h \in V_{cG}(\Omega). \end{aligned} \tag{6.2.11}$$

where q_h^I, q_h^P *and* $\hat{c}_h$ *are the* L^2 *projections of* q^I, q^P *and* $\hat{c}$ *onto the approximation space.*

Note that we now find the approximation for velocity via $u_h = -a(c_h)\nabla p_h$.

Definition 6.2.12. *Define the auxiliary problem as: Find* $(\tilde{c}, \tilde{p}) \in W^{1,\infty}(\Omega) \times W_0$ *such that*

$$\begin{aligned} \Phi(\tilde{p}; c_h) &= q^I - q^P, \\ \Psi(\tilde{c}; p_h) &= \hat{c} q^I \end{aligned} \tag{6.2.13}$$

where (c_h, p_h) *is the solution to the discrete problem.*

For simplicity we specify that $q^I = q_h^I$, $q^P = q_h^P$ and $\hat{c} = \hat{c}_h$.

We now follow the steps of the abstract analysis for this problem. The methods we use for the a posteriori estimators (Assumption 6.1.5) can be found in, e.g., [5], although many alternatives exist in the literature. The coupling assumption (Assumption 6.1.4) follows using standard techniques which can be found in most finite element analysis texts, e.g., [34, 67]. The primary purpose of presenting this analysis therefore is not to prove that the stationary problem (6.2.1)-(6.2.3) fits into the abstract framework as this is relatively straightforward. The main purpose is to show the process in order to understand the restrictions we face in non-convex domains and the properties we will have to duplicate (to follow this approach) in weighted spaces.

We first show coercivity of Φ and Ψ when the coefficients of the problem satisfy the assumptions of Section 4.2. Recall $a = \mathbb{K}/\mu$ so using Assumption (A1) and Assumption (A2) we have $a_\circ \leq a \leq a^\circ$ for positive $a_\circ, a^\circ \in \mathbb{R}$. For $p \in W_0$ we have

$$\begin{aligned} \Phi(p; c)[p] &= (a(c)\nabla p, \nabla p) \\ &\geq a_\circ \|\nabla p\|^2_{L^2(\Omega)} \\ &\geq \Xi_p \|p\|^2_{H^1(\Omega)} \end{aligned} \tag{6.2.14}$$

where in the final line we have used a Poincaré inequality and so Ξ_p depends on the constant there and $a_\circ$. Using Assumptions (A3) and (A5) in a similar manner for

$c \in H^1(\Omega)$ we find

$$\begin{aligned}\Psi(c;p)[c] &= (\mathbb{D}(u)\nabla c, \nabla c) + \frac{1}{2}(u\cdot\nabla c, c) - \frac{1}{2}(uc, \nabla c) + \frac{1}{2}((q^I + q^P)c, c)\\ &= (\mathbb{D}(u)\nabla c, \nabla c) + (\frac{1}{2}(q^I + q^P)c, c)\\ &\geq d_\circ \|\nabla c\|^2_{L^2(\Omega)} + \frac{1}{2}\min(q^I + q^P)\|c\|^2_{L^2(\Omega)}\\ &\geq \Xi_c \|c\|^2_{H^1(\Omega)}.\end{aligned} \tag{6.2.15}$$

Note that if $\min(q^I + q^P) = 0$ (as is usual) we require a Poincaré inequality as for pressure.

Lemma 6.2.16. *Let $(c,p) \in W^{1,\infty}(\Omega)\times W_0$ be the solution to the continuous problem defined in Definition 6.2.8, and $(\tilde{c},\tilde{p}) \in W^{1,\infty}(\Omega)\times W_0$ be the solution to the auxiliary problem defined in Definition 6.2.12. Then we have*

$$\|p - \tilde{p}\|^2_{H^1(\Omega)} \leq \gamma_p \|c - c_h\|^2_{H^1(\Omega)} \tag{6.2.17}$$

where

$$\gamma_p = \left(\frac{a^\circ \|\nabla p\|_{L^\infty(\Omega)}}{\Xi_p}\right)^2, \tag{6.2.18}$$

and

$$\|c - \tilde{c}\|^2_{H^1(\Omega)} \leq \gamma_c \|p - p_h\|^2_{H^1(\Omega)} \tag{6.2.19}$$

where

$$\gamma_c = \left(\frac{a^\circ\left((d^\circ + \frac{1}{2})\|\nabla c\|_{L^\infty(\Omega)} + \frac{1}{2}\|c\|_{L^\infty(\Omega)}\right)}{\Xi_c}\right)^2. \tag{6.2.20}$$

Proof. By subtracting the Φ terms in (6.2.9) from those in (6.2.13) we find $\Phi(\tilde{p};c_h)[w] - \Phi(p;c)[w] = 0$ for $w \in W_0(\Omega)$. Using this result with the coercivity from (6.2.14)

gives

$$\begin{aligned}\Xi_p\|p-\tilde{p}\|^2_{H^1(\Omega)} &\leq \Phi(p-\tilde{p};c_h)[p-\tilde{p}]\\ &= \Phi(p;c_h)[p-\tilde{p}]-\Phi(p;c)[p-\tilde{p}]-\Phi(\tilde{p};c_h)[p-\tilde{p}]+\Phi(p;c)[p-\tilde{p}]\\ &= ((a(c_h)-a(c))\nabla p,\nabla(p-\tilde{p}))\\ &\leq a^\circ\|c-c_h\|_{H^1(\Omega)}\|\nabla p\|_{L^\infty(\Omega)}\|p-\tilde{p}\|_{H^1(\Omega)}\end{aligned}$$

and by rearrangement we have shown (6.2.17).

Similarly by subtracting the Ψ terms in (6.2.9) from those in (6.2.13) we have $\Psi(c;p)[d]-\Psi(\tilde{c};p_h)[d]=0$ for $d\in W^{1,\infty}(\Omega)$. Then using the coercivity from (6.2.15) we have

$$\begin{aligned}&\Xi_c\|c-\tilde{c}\|^2_{H^1(\Omega)}\\ &\leq \Psi(c-\tilde{c};p_h)[c-\tilde{c}]\\ &= \Psi(c;p_h)[c-\tilde{c}]-\Psi(c;p)[c-\tilde{c}]-\Psi(\tilde{c};p_h)[c-\tilde{c}]+\Psi(c;p)[c-\tilde{c}]\\ &= (\mathbb{D}(u_h)\nabla c-\mathbb{D}(u)\nabla c,\nabla(c-\tilde{c}))+\frac{1}{2}(u_h\cdot\nabla c-u\cdot\nabla c,c-\tilde{c})\\ &\quad -\frac{1}{2}(u_hc-uc,\nabla(c-\tilde{c}))\\ &\leq \|\mathbb{D}(u)-\mathbb{D}(u_h)\|_{L^2(\Omega)}\|\nabla c\|_{L^\infty(\Omega)}\|c-\tilde{c}\|_{H^1(\Omega)}\\ &\quad +\frac{1}{2}\|u-u_h\|\|c-\tilde{c}\|_{H^1(\Omega)}\left(\|\nabla c\|_{L^\infty(\Omega)}+\|c\|_{L^\infty(\Omega)}\right)\\ &\leq a^\circ\left(d^\circ\|\nabla c\|_{L^\infty(\Omega)}+\frac{1}{2}\|\nabla c\|_{L^\infty(\Omega)}+\frac{1}{2}\|c\|_{L^\infty(\Omega)}\right)\|p-p_h\|_{H^1(\Omega)}\|c-\tilde{c}\|_{H^1(\Omega)}\end{aligned}$$

where we have used (6.2.3) and $u_h=-a(c_h)\nabla p_h$. By rearrangement we have (6.2.20). □

The γ_c and γ_p found here may not be optimum. It may be necessary to formulate sharper constants to satisfy $\gamma_c\gamma_p<1$ (cf., Assumption 6.1.4). For the purposes of this exposition we simply assume this is the case for the constants calculated above.

Assumption 6.2.21. *We assume that γ_c and γ_p defined in Lemma 6.2.16 satisfy $\gamma_c\gamma_p<1$.*

We now present simple a posteriori error estimators in the manner of Assumption

6.1.5. We remark that the literature in this area is very well developed and the estimators presented here are by no means original or unique. We present them for completeness and to motivate later sections. For alternative estimators see, e.g., [4, 17, 31] (this is not intended to be a comprehensive list).

Lemma 6.2.22. *Let $(\tilde{c}, \tilde{p}) \in W^{1,\infty}(\Omega) \times W_0(\Omega)$ be the solution of (6.2.13), and let $(c_h, p_h) \in V_{cG} \times (V_{cG} \cap W_0(\Omega))$ be the approximation defined in (6.2.11). With $q^I = q_h^I$, $q^P = q_h^P$ and $\hat{c} = \hat{c}_h$ we have the following a posteriori error estimators:*

$$\begin{aligned} \|\tilde{p} - p_h\|^2_{H^1(\Omega)} \leq C_p \Bigg(& \sum_{E \in \mathcal{T}_h} h_E^2 \|q^I - q^P + \nabla \cdot (a(c_h) \nabla p_h)\|^2_{L^2(E)} \\ & + \sum_{e \in \mathcal{E}_h^o} h_e \|[\![a(c_h) \nabla p_h]\!]\|^2_{L^2(e)} \Bigg) \end{aligned} \tag{6.2.23}$$

and

$$\begin{aligned} & \|\tilde{c} - c_h\|^2_{H^1(\Omega)} \\ & \leq C_c \left(\sum_{E \in \mathcal{T}_h} h_E^2 \|\mathtt{R_c}\|^2_{L^2(E)} + \sum_{e \in \mathcal{E}_h^o} h_e \|[\![\mathbb{D}(u_h) \nabla c_h]\!] + \frac{1}{2} [\![u_h c_h]\!]\|^2_{L^2(e)} \right) \end{aligned} \tag{6.2.24}$$

where

$$\mathtt{R_c} = \hat{c} q^I - \nabla \cdot (\mathbb{D}(u_h) \nabla c_h) - \frac{1}{2} u_h \cdot \nabla c_h + \frac{1}{2} \nabla \cdot (u_h c_h) - \frac{1}{2} (q^I + q^P) c_h. \tag{6.2.25}$$

Here C_p depends on the coercivity coefficient in (6.2.14) and C_{sz} in (3.3.3), and C_c depends on the coercivity coefficient in (6.2.15) and C_{sz}.

Proof. Subtracting the Φ terms in (6.2.13) from those in (6.2.11) we have the Galerkin orthogonality

$$\Phi(\tilde{p} - p_h; c_h)[w_h] = 0 \qquad \forall w_h \in V_{\mathrm{cG}} \cap W_0(\Omega). \tag{6.2.26}$$

Integration by parts on the weak form gives

$$\begin{aligned}
&\Phi(\tilde{p} - p_h; c_h)[w] \\
&= (a(c_h)\nabla(\tilde{p} - p_h), \nabla w) \\
&= (q^I - q^P, w) - (a(c_h)\nabla p_h, \nabla w) \\
&= (q^I - q^P, w) + \sum_{E \in \mathcal{T}_h} (\nabla \cdot (a(c_h)\nabla p_h), w)_E - \sum_{e \in \mathcal{E}_h^o} \int_e w [\![a(c_h)\nabla p_h]\!] \, \mathrm{d}s
\end{aligned}$$

where we have applied (1.3.25). By adding (6.2.26) with $w_h = \mathcal{SZ}_h(w)$, the Scott-Zhang projection introduced in Chapter 3, we find

$$\begin{aligned}
\Phi(\tilde{p} - p_h; c_h)[w] &= \Phi(\tilde{p} - p_h; c_h)[w - \mathcal{SZ}_h(w)] \\
&= \sum_{E \in \mathcal{T}_h} (q^I - q^P + \nabla \cdot (a(c_h)\nabla p_h), w - \mathcal{SZ}_h(w))_E \\
&\quad - \sum_{e \in \mathcal{E}_h^o} \int_e [\![a(c_h)\nabla p_h]\!] (w - \mathcal{SZ}_h(w)) \\
&\leq \sum_{E \in \mathcal{T}_h} \| q^I - q^P + \nabla \cdot (a(c_h)\nabla p_h) \|_{L^2(E)} \| w - \mathcal{SZ}_h(w) \|_{L^2(E)} \\
&\quad + \sum_{e \in \mathcal{E}_h^o} \| [\![a(c_h)\nabla p_h]\!] \|_{L^2(e)} \| w - \mathcal{SZ}_h(w) \|_{L^2(e)}.
\end{aligned}$$

By setting $w = \tilde{p} - p_h$ and using the coercivity of Φ from (6.2.14) and the properties of the Scott-Zhang projection (3.3.3) we show (6.2.23).

For the Ψ terms of (6.2.13) and (6.2.11) by subtraction we have the Galerkin orthogonality

$$\Psi(\tilde{c} - c_h; p_h)[d_h] = 0 \quad \forall d_h \in V_{\mathrm{cG}}. \tag{6.2.27}$$

Using (6.2.7) and integrating we find

$$\begin{aligned}
&\Psi(\tilde{c}-c_h;p_h)[d]\\
&=(\hat{c}q^I,d)-(\mathbb{D}(u_h)\nabla c_h,\nabla d)-\frac{1}{2}(u_h\cdot\nabla c_h,d)+\frac{1}{2}(u_hc_h,\nabla d)-\frac{1}{2}((q^I+q^P)c_h,d)\\
&=(\hat{c}q^I,d)+\sum_{E\in\mathcal{T}_h}(\nabla\cdot(\mathbb{D}(u_h)\nabla c_h)-\frac{1}{2}u_h\cdot\nabla c_h+\frac{1}{2}\nabla\cdot(u_hc_h)-\frac{1}{2}(q^I+q^P)c_h,d)_E\\
&\quad-\sum_{e\in\mathcal{E}_h^o}([\![\mathbb{D}(u_h)\nabla c_h]\!]+\frac{1}{2}[\![u_hc_h]\!],d)_e.
\end{aligned}$$

Combining this with the Galerkin orthogonality and $d_h=\mathcal{SZ}_h(d)$ we find

$$\begin{aligned}
\Psi(\tilde{c}-c_h;p_h)[d]&=\Psi(\tilde{c}-c_h;p_h)[d-\mathcal{SZ}_h(d)]\\
&=\sum_{E\in\mathcal{T}_h}(\mathtt{R_c},d-\mathcal{SZ}_h(d))\\
&\quad-\sum_{e\in\mathcal{E}_h^o}([\![\mathbb{D}(u_h)\nabla c_h]\!]+\frac{1}{2}[\![u_hc_h]\!],d-\mathcal{SZ}_h(d))_e\\
&\le\sum_{E\in\mathcal{T}_h}\|\mathtt{R_c}\|_{L^2(E)}\|d-\mathcal{SZ}_h(d)\|_{L^2(E)}\\
&\quad+\sum_{e\in\mathcal{E}_h^o}\|[\![\mathbb{D}(u_h)\nabla c_h]\!]+\frac{1}{2}[\![u_hc_h]\!]\|_{L^2(e)}\|d-\mathcal{SZ}_h(d)\|_{L^2(e)}.
\end{aligned}$$

By choosing $d=\tilde{c}-c_h$, using the coercivity of Ψ in (6.2.15) and the properties of the Scott-Zhang projection we recover (6.2.24). □

Theorem 6.2.28. *Let γ_p and γ_c defined in (6.2.18) and (6.2.20) resp. satisfy Assumption 6.2.21. Then the solution $(c,p)\in W^{1,\infty}(\Omega)\times W_0(\Omega)$ to (6.2.9) and $(c_h,p_h)\in V_{cG}\times(V_{cG}\cap W_0(\Omega))$ defined by (6.2.11) satisfy the a posteriori error estimate*

$$\begin{aligned}
&\|p-p_h\|_{H^1(\Omega)}^2+\|c-c_h\|_{H^1(\Omega)}^2\\
&\le C\left(\sum_{E\in\mathcal{T}_h}h_E^2\left(\|q^I-q^P+\nabla\cdot(a(c_h)\nabla p_h)\|_{L^2(E)}^2+\|\mathtt{R_c}\|_{L^2(E)}^2\right)\right.\\
&\quad\left.+\sum_{e\in\mathcal{E}_h^o}h_e\left(\|[\![a(c_h)\nabla p_h]\!]\|_{L^2(e)}^2+\|[\![\mathbb{D}(u_h)\nabla c_h]\!]+\frac{1}{2}[\![u_hc_h]\!]\|_{L^2(e)}^2\right)\right)
\end{aligned}\tag{6.2.29}$$

where C depends on the constants defined in Lemmas 6.2.16 and 6.2.22.

Proof. Following the steps of the proof of Theorem 6.1.8 we combine the results of Lemmas 6.2.16 and 6.2.22. □

6.3 The Case for an Alternative Approach

If $\Omega \subset \mathbb{R}^d$ is a smooth, bounded domain then the solution to a strongly elliptic differential equation will be smooth on $\overline{\Omega}$ provided the given data and coefficients are smooth. This is a well known result which was referenced in Section 4.3 and is used to show the a posteriori estimators in Chapter 5. In non smooth/non convex domains there can be the loss of regularity and the smoothness of the data no longer implies the smoothness of the solution.

Consider for example the case of an L-shaped domain in $\mathbb{R}^2$ as studied in [20, Numerical Example 2] and shown in Figure 6.3.1. For injection and production wells located at $(1, 1)$ and $(0, 0)$ respectively, and with discontinuous permeability, we see a singular velocity field at $(0.5, 0.5)$, the location of the re-entrant corner.

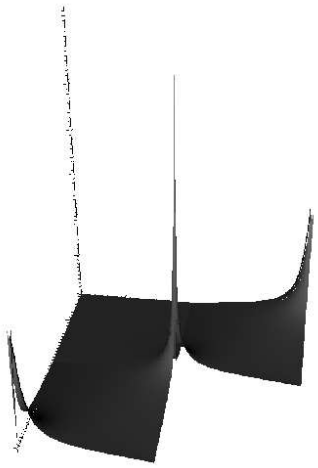


Figure 6.3.1: A numerical simulation showing an approximation to the unbounded velocity field on an L-shaped domain with a re-entrant corner at $(0.5, 0.5)$ and singular injection/extraction points at $(0, 0)$ and $(1, 1)$ respectively.

We will outline an approach we believe can be used to address some of the difficulties associated in generating a posteriori error estimators for such problems

using the so called weighted (Babuška-Kondratiev) spaces as described in Section 6.4. Our analysis is incomplete and our intention is to highlight the difficulties still faced while presenting some useful results which may be applied when dealing with weighted spaces.

We face several problems in trying to extend the analysis of the previous section to problems in non convex domains.

- **Existence and uniqueness**. Existence and uniqueness in the weighted spaces has been shown for a class of elliptic equations [100, 101]. However no results exist for the coupled problem (6.2.1)-(6.2.2) in non convex domains using weighted spaces. Extension to parabolic problems has been considered by, e.g., [28, 29, 72, 92]. No results in weighted spaces exist for the coupled problem (1.1.3)-(1.1.5).

- **Coercivity**. It is not clear that coercivity results of the type (6.2.14) and (6.2.15) exist in the weighted spaces. The coercivity was used to show both the coupling assumption and the a posteriori error estimators (Lemmas 6.2.16 and 6.2.22) in Section 6.2, and as such it is a central tool in the analysis.

- **Boundedness of derivatives**. We do not have $c, p \in W^{1,\infty}(\Omega)$ (see for example Figure 6.3.1). Therefore we cannot employ the methods leading to, e.g., (6.2.18) and (6.2.20). Alternative bounds must be formulated in the weighted spaces.

In the following sections we concentrate on the final item and show that if we assume existence of a solution to (6.2.1)-(6.2.2) in some weighted space we may generate an a posteriori error estimator for the stationary miscible displacement problem when Ω is not convex.

6.4 Some Results from the Theory of Weighted Spaces

We consider only problems where $\Omega \in \mathbb{R}^2$ is a straight sided polygonal domain. We do not allow cracks and assume that the domain is Lipschitz. In particular we

consider problems with non smooth points on the boundary arising from corners and from coefficients which are singular on the boundary. Both may give rise to point type singularities, i.e., points where the solution or its derivative are singular. We call these the *collection of singular points*.

For a general introduction to weighted spaces see the books by Kufner [93] or Dauge [58]. Work on weighted spaces covers a variety of areas and an exhaustive survey is not practical here. In the elliptic case we suggest that readers study [6, 13] and the references contained therein. Less work has been undertaken for the parabolic case. We refer readers to [28, 29, 72, 92] in particular. Our analysis in the elliptic case will follow the notation of [6, 100, 101]. The work of [100, 101] shows that we have only weighted control on the H^2 semi-norm on regions such as the L shaped domain above. However we have greater control on the lower order norms.

We introduce some results from the theory of weighted spaces following the notation of Ammann and Nistor [6]. The collection of singular points Q is denoted by $\mathcal{V}$, also called the vertices. If $r_Q(\boldsymbol{x})$ is the distance from $\boldsymbol{x}$ to $Q \in \mathcal{V}$ (using paths in $\overline{\Omega}$) then define the *weight function* by

$$\vartheta(\boldsymbol{x}) = \prod_{Q \in \mathcal{V}} r_Q(\boldsymbol{x}).$$

Let $\vec{a} = (a_Q)$ be a vector with real components indexed by $Q \in \mathcal{V}$. For $t \in \mathbb{R}$ denote $\vec{a} + t = (a_Q + t)$ and note $\vartheta^{\vec{a}+t}(\boldsymbol{x}) = \vartheta^{\vec{a}}(\boldsymbol{x})\vartheta^{t}(\boldsymbol{x})$. If we write, e.g., $\vec{a} = s$, $s \in \mathbb{R}$ we mean $\vec{a} = (s)$, a vector with length corresponding to the number of vertices with each entry being s. Similarly by writing, e.g., $\vec{a} + 1$ we denote the addition of 1 elementwise to $\vec{a}$. We assume that $0 \leq \vec{a} \leq 1$ unless specifically stated otherwise.

Definition 6.4.1. *We define the weighted Sobolev (Babuška-Kondratiev) space to be*

$$\mathcal{K}^m_{\vec{a}}(\Omega) := \{f : \vartheta^{|\alpha|-\vec{a}} D^\alpha f \in L^2(\Omega),\ \forall |\alpha| \leq m \in \mathbb{Z}^+\} \tag{6.4.2}$$

where $\mathbb{Z}^+$ is the set of non-negative integers.

Note that $L^2(\Omega) = \mathcal{K}^0_0(\Omega)$ but that, e.g., $H^1(\Omega) \neq \mathcal{K}^1_0(\Omega)$ as the power of ϑ

depends on the order of the derivative. The relationship between the weighted and unweighted Sobolev spaces is key to our analysis.

Define the norm on the Babuška-Kondratiev space to be

$$\|v\|^2_{\mathcal{K}^m_{\vec{a}}(\Omega)} = \sum_{|\alpha|\leq m} \|\vartheta^{|\alpha|-\vec{a}} D^\alpha v\|^2_{L^2(\Omega)} \tag{6.4.3}$$

and the inner product

$$(u,v)_{\mathcal{K}^m_{\vec{a}}(\Omega)} = \sum_{|\alpha|\leq m} \int_\Omega \vartheta^{2(|\alpha|-\vec{a})} (D^\alpha u)(D^\alpha v)\,\mathrm{d}\boldsymbol{x}. \tag{6.4.4}$$

The spaces $\mathcal{K}^m_{\vec{a}}(\partial\Omega)$ on the boundary are defined similarly, i.e., where $\mathcal{P}$ is a differential operator of positive integer order k on $\partial\Omega$ we have

$$\mathcal{K}^m_{\vec{a}}(\partial\Omega) := \{f : \partial\Omega \to \mathbb{R}, \vartheta^{k-\vec{a}}\mathcal{P}(f|_{\partial\Omega}) \in L^2(\partial\Omega),\ \forall m \in \mathbb{Z}^+\}$$

and

$$\|v\|^2_{\mathcal{K}^m_{\vec{a}}(\partial\Omega)} = \sum_{k\leq m} \|\vartheta^{k-\vec{a}}\mathcal{P}v\|^2_{L^2(\partial\Omega)}.$$

We present the following lemma from [6, Theorem 3.8].

Lemma 6.4.5. *On a polygonal domain Ω we have that the restriction to the boundary extends to a continuous, surjective map*

$$\mathcal{K}^m_{\vec{a}}(\Omega) \ni u \to u|_{\partial\Omega} \in \mathcal{K}^{m-1/2}_{\vec{a}-1/2}(\partial\Omega) \tag{6.4.6}$$

for $m \geq 1$.

A consequence of this result is that there exists a positive constant C, depending on m and $\vec{a}$, such that

$$\|v\|_{\mathcal{K}^{m-1/2}_{\vec{a}-1/2}(\partial\Omega)} \leq C\|v\|_{\mathcal{K}^m_{\vec{a}}(\Omega)} \qquad \forall v \in \mathcal{K}^m_{\vec{a}}(\Omega). \tag{6.4.7}$$

Consider the second order differential operator

$$\mathcal{L} = -\sum_{i,j=1}^{2} \partial_{x_j} A^{ij} \partial_{x_i} + \sum_{i=1}^{2} b^i \partial_{x_i} + c \tag{6.4.8}$$

where $A^{ij} = A^{ji}$. We assume that we have strong ellipticity, namely,

$$\sum_{i,j=1}^{2} A^{ij}(x)\xi_i\xi_j \geq C\|\xi\|^2_{L^2(\Omega)} \tag{6.4.9}$$

for some constant $C > 0$ independent of $x \in \overline{\Omega}$ and $\xi \in \mathbb{R}^2$. We also assume as previously that $c - \frac{1}{2}\nabla \cdot b > 0$. We shall assume that the coefficients have singularities only on the boundary but are otherwise smooth. Consider now the boundary value problem

$$\begin{aligned} \mathcal{L}p &= f \qquad \text{in } \Omega, \\ p &= g_D \qquad \text{on } \partial\Omega_D, \\ \nabla p \cdot n &= g_N \qquad \text{on } \partial\Omega_N, \end{aligned} \tag{6.4.10}$$

where the boundary $\partial\Omega$ is decomposed into Dirichlet and Neumann components denoted by $\partial\Omega_D$ and $\partial\Omega_N$ respectively, $\partial\Omega_D \cap \partial\Omega_N = \emptyset$ and $\partial\Omega_D \neq \emptyset$. We say this problem has singular points only on the boundary and that $\vec{a}$ is known. By [101, Theorem 3.2] there exists a positive constant η_Q such that for all $|a_Q| < \eta_Q$ there exists a unique weak solution $p \in \mathcal{K}^1_{\vec{a}+1}(\Omega)$ with $p = 0$ on $\partial\Omega_D$. Then we have the following theorem concerning the regularity of the solution p of (6.4.10) which is a simplification of that in [101].

Theorem 6.4.11. *Let $m \geq 1$. Assume that $g_N \in \mathcal{K}^{m-1/2}_{\vec{a}-1/2}(\partial\Omega_N)$, $g_D \in \mathcal{K}^{m+1/2}_{\vec{a}+1/2}(\partial\Omega_D)$ and $f \in \mathcal{K}^{m-1}_{\vec{a}-1}(\Omega)$ for some straight sided polygonal domain $\Omega \in \mathbb{R}^2$ with finitely many singular points Q only on the boundary, each with associated paramater $0 \leq a_Q \leq 1$. Then for $|a_Q| < \eta_Q$ a positive constant we have that the solution $p \in \mathcal{K}^1_{\vec{a}+1}(\Omega)$ to (6.4.10) satisfies $p \in \mathcal{K}^{m+1}_{\vec{a}+1}(\Omega)$ and we have the estimate*

$$\|p\|_{\mathcal{K}^{m+1}_{\vec{a}+1}(\Omega)} \leq C_{BK}\left(\|f\|_{\mathcal{K}^{m-1}_{\vec{a}-1}(\Omega)} + \|g_N\|_{\mathcal{K}^{m-1/2}_{\vec{a}-1/2}(\partial\Omega_N)} + \|g_D\|_{\mathcal{K}^{m+1/2}_{\vec{a}+1/2}(\partial\Omega_N)}\right) \tag{6.4.12}$$

where C_{BK} is a constant independent of f, g_N and g_D but perhaps depends on $\vec{a}$.

In particular when $m = 1$ we have $p \in \mathcal{K}^2_{\vec{a}+1}(\Omega)$. If we also have $\vec{a} = 1$ the highest derivative in the weighted Sobolev space is unweighted, i.e., from (6.4.2) we have $|\alpha| - (\vec{a} + 1) = 0$ for $|\alpha| = 2$. Less regular domains have smaller a_Q and when $\vec{a} < 1$ the power of ϑ on the highest derivative is positive, i.e., $|\alpha| - (\vec{a} + 1) > 0$ for $|\alpha| = 2$. However for $|\alpha| = 0, 1$ the power of ϑ is negative, e.g., $\vartheta^b p \in L^2(\Omega)$ for $b < 0$, and similarly for the derivative. It is this property which we will exploit in the next section.

6.5 Sobolev Imbeddings in Weighted Spaces

We wish to extend some results of Sobolev imbeddings to weighted spaces. Our approach here is restricted to our particular problem and readers should refer to, e.g., [40, 76] and the references therein for more general results on the imbedding of weighted Sobolev spaces. Note that we assume that we have a sufficiently regular boundary to apply these imbeddings, i.e., a Lipschitz boundary.

We first present a lemma (modifying the notation) from [13].

Lemma 6.5.1. *Let $\vec{a}$ be as defined previously, $b \in \mathbb{R}$ and let $\mathcal{L}$ be a differential operator of order k with smooth coefficients. Then for $m \in \mathbb{Z}$ we have that multiplication by ϑ^b defines an isomorphism $\mathcal{K}^m_{\vec{a}}(\Omega) \to \mathcal{K}^m_{\vec{a}+b}(\Omega)$. Thus $\vartheta^b \mathcal{K}^m_{\vec{a}}(\Omega) = \mathcal{K}^m_{\vec{a}+b}(\Omega)$ where $\vartheta^b \mathcal{K}^m_{\vec{a}}(\Omega) = \{\vartheta^b v, \forall v \in \mathcal{K}^m_{\vec{a}}(\Omega)\}$. In addition the map $\mathcal{L} : \mathcal{K}^m_{\vec{a}}(\Omega) \to \mathcal{K}^{m-k}_{\vec{a}-k}(\Omega)$ is well defined and continuous.*

The proof of this lemma can be found in [6]. Useful in the proof is the additional result from [100].

Lemma 6.5.2. *For positive $i, j \in \mathbb{Z}$ the function $\vartheta^{j+i-\vec{a}} \partial_x^j \partial_y^i \vartheta^{\vec{a}}$ is bounded on Ω.*

A consequence of Lemma 6.5.1 is that any first order derivatives of functions in a given weighted Sobolev space will be in a weighted Sobolev space of one order lower in each index, e.g., functions in $\mathcal{K}^m_{\vec{a}}(\Omega)$ have derivatives in $\mathcal{K}^{m-1}_{\vec{a}-1}(\Omega)$. We will also

frequently use the fact that for $m \geq m'$ and $\vec{a} \geq \vec{a}'$ we have in bounded domains

$$\mathcal{K}^m_{\vec{a}}(\Omega) \subset \mathcal{K}^{m'}_{\vec{a}'}(\Omega).$$

In particular this means that $\mathcal{K}^0_{\vec{a}}(\Omega) \subset L^2(\Omega)$ for $\vec{a} \geq 0$ and $\mathcal{K}^1_{\vec{a}+1}(\Omega) \subset H^1(\Omega)$ for $\vec{a}+1 \geq 1$. We will use these results without justification throughout the text.

Recall the following Sobolev embedding theorem from, e.g., [3, Theorem 4.12].

Theorem 6.5.3. *Let Ω be a domain in $\mathbb{R}^2$ satisfying the cone condition [3, Definition 4.6]. Let $j \geq 0$ and $m \geq 1$ be integers and let $1 \leq p \leq \infty$. If $mp > 2$ then*

$$W^{j+m,p}(\Omega) \to W^{j,q}(\Omega) \qquad \text{for } p \leq q \leq \infty,$$

and in particular

$$H^2(\Omega) \to L^q(\Omega) \qquad \text{for } 2 \leq q \leq \infty \tag{6.5.4}$$

when $m = 2$, $p = 2$. If $mp = 2$ then

$$W^{j+m,p}(\Omega) \to W^{j,q}(\Omega) \qquad \text{for } p \leq q < \infty,$$

and in particular

$$H^1(\Omega) \to L^q(\Omega) \qquad \text{for } 2 \leq q < \infty \tag{6.5.5}$$

when $m = 1$, $p = 2$.

We want to relate these Sobolev imbeddings to weighted spaces. We begin by asking: Given $w \in \mathcal{K}^2_{\vec{a}+1}(\Omega)$ how should we choose the real valued vector b so that $\vartheta^b w \in H^2(\Omega)$?

Lemma 6.5.6. *If $w \in \mathcal{K}^2_{\vec{a}+1}(\Omega)$ then $\vartheta^{1-\vec{a}} w \in H^2(\Omega)$.*

Proof. As $w \in \mathcal{K}^2_{\vec{a}+1}(\Omega)$ we have immediately that $\vartheta^{1-\vec{a}} w \in L^2(\Omega)$. From Lemma

6.5.1 we have $\vartheta^{1-\vec{a}}w \in \mathcal{K}^2_2(\Omega)$. Using the definition of the weighted spaces

$$\|\vartheta^{1-\vec{a}}w\|^2_{\mathcal{K}^2_2(\Omega)} = \|\vartheta^{-1-\vec{a}}w\|^2_{L^2(\Omega)} + \|\vartheta^{-1}D^1(\vartheta^{1-\vec{a}}w)\|^2_{L^2(\Omega)} + \|D^2(\vartheta^{1-\vec{a}}w)\|^2_{L^2(\Omega)}.$$

Thus we see $D^2(\vartheta^{1-\vec{a}}w) \in L^2(\Omega)$ and $D(\vartheta^{1-\vec{a}}w) \in L^2(\Omega)$. Each derivative of $\vartheta^{1-\vec{a}}w$ is in $L^2(\Omega)$ and so $\vartheta^{1-\vec{a}}w \in H^2(\Omega)$. □

Lemma 6.5.7. *For $\Omega \subset \mathbb{R}^2$ satisfying the conditions of Theorem 6.5.3 if $w \in \mathcal{K}^2_{\vec{a}+1}(\Omega)$ then $\vartheta^{1-\vec{a}}w \in L^\infty(\Omega)$.*

Proof. By Lemma 6.5.6 we have $\vartheta^{1-\vec{a}}w \in H^2(\Omega)$. Then by using (6.5.4) we have $\vartheta^{1-\vec{a}}w \in L^\infty(\Omega)$. □

Of course we should not expect that the gradient of $w \in \mathcal{K}^2_{\vec{a}+1}(\Omega)$ will be bounded. However we can show by a similar method that the weighted gradient is in a Lebesgue space.

Lemma 6.5.8. *For $\Omega \subset \mathbb{R}^2$ satisfying the conditions of Theorem 6.5.3 if $w \in \mathcal{K}^2_{\vec{a}+1}(\Omega)$ then $\vartheta^{1-\vec{a}}\nabla w \in L^q(\Omega)$ for $2 \leq q < \infty$.*

Proof. By Lemma 6.5.6 we have $\nabla(\vartheta^{1-\vec{a}}w) \in H^1(\Omega)$. Using Lemma 6.5.2 gives $\vartheta^{1-\vec{a}}\nabla w \in H^1(\Omega)$. Then using (6.5.5) we find $\vartheta^{1-\vec{a}}\nabla w \in L^q(\Omega)$ for $2 \leq q < \infty$. □

Note that this result is not the weighted analogue of $\nabla w \in L^\infty(\Omega)$ for solutions of elliptic equations in convex domains, i.e., for $w \in H^2(\Omega)$ the Sobolev embedding gives $\nabla w \in L^q(\Omega)$ for $2 \leq q < \infty$ but the approach of, e.g., [102] is used to show $\nabla w \in L^\infty(\Omega)$. We have not been able to prove a result of the type $\vartheta^b \nabla w \in L^\infty(\Omega)$ for $w \in \mathcal{K}^2_{\vec{a}+1}(\Omega)$ for some b. If such a result could be shown it would be a very useful contribution to the field.

6.6 A Posteriori Error Estimators in the Weighted Spaces

In order to formulate a posteriori error estimators in the weighted spaces following the abstract approach of Section 6.1 we must make some assumptions on the solution to (6.2.1)-(6.2.2).

Assumption 6.6.1. *Let $\Omega \in \mathbb{R}^2$ be a bounded straight edged polygonal domain without cracks in which the Sobolev embeddings of Section 6.5 may be applied. Then we assume that the problem (6.2.1)-(6.2.5) has singular points only on the boundary and a unique solution $(c,p) \in \mathcal{K}^2_{\vec{a}}(\Omega) \times \mathcal{K}^{2,0}_{\vec{a}}(\Omega)$, where $\mathcal{K}^{2,0}_{\vec{a}}(\Omega) := \{w \in \mathcal{K}^2_{\vec{a}}(\Omega) : \int_\Omega w \,\mathrm{d}\boldsymbol{x} = 0\}$. Furthermore we assume we have the bound (6.4.12) with $m = 1$ for each of c and p with the boundary conditions as given.*

As we have remarked to our knowledge there are no results concerning the solution to the coupled stationary IMD problem in the weighted spaces. The assumption we make is therefore based on the regularity for the elliptic problem presented in Section 6.4, with a condition on the pressure to ensure unique solutions.

The weak forms of (6.2.1) and (6.2.2) are given by, for all $w \in \mathcal{K}^{1,0}_{\vec{a}+1}(\Omega)$ and $d \in \mathcal{K}^1_{\vec{a}+1}(\Omega)$,

$$\Phi(p;c)[w] = (w \mapsto (a(c)\nabla p, \nabla w)), \tag{6.6.2}$$

$$\Psi(c;p)[d] = (d \mapsto (\mathbb{D}(u)\nabla c, \nabla d) + \frac{1}{2}(u \cdot \nabla c, d) - \frac{1}{2}(uc, \nabla d) + \frac{1}{2}((q^I + q^P)c, d)). \tag{6.6.3}$$

We must also make an assumption about the coercivity of the operators.

Assumption 6.6.4. *The operators Φ and Ψ defined above are coercive in $\mathcal{K}^1_{\vec{a}+1}(\Omega)$, i.e.,*

$$\Phi(p;c)[p] \geq \Xi_p \|p\|^2_{\mathcal{K}^1_{\vec{a}+1}(\Omega)} \tag{6.6.5}$$

and

$$\Psi(c;p)[c] \geq \Xi_c \|c\|^2_{\mathcal{K}^1_{\vec{a}+1}(\Omega)}. \tag{6.6.6}$$

We define the continuous, discrete and auxiliary problem (cf., (6.1.1), (6.1.2) and (6.1.3)):

Definition 6.6.7. *Define the continuous problem as: Find $(c,p) \in \mathcal{K}^1_{\vec{a}+1}(\Omega) \times$*

$\mathcal{K}^{1,0}_{\vec{a}+1}(\Omega)$ *such that*

$$\begin{aligned} \Phi(p;c) &= q^I - q^P, \\ \Psi(c;p) &= \hat{c}q^I. \end{aligned} \tag{6.6.8}$$

Definition 6.6.9. *Define the discrete problem as: Find* $(c_h, p_h) \in V_{cG} \times V_{cG} \cap L^2_0(\Omega)$ *such that*

$$\begin{aligned} \Phi(p_h;c_h)[w_h] &= q^I_h - q^P_h \qquad \forall w_h \in V_{cG}(\Omega) \cap L^2_0(\Omega), \\ \Psi(c_h;p_h)[d_h] &= \hat{c}_h q^I_h \qquad \forall d_h \in V_{cG}(\Omega). \end{aligned} \tag{6.6.10}$$

Definition 6.6.11. *Define the auxiliary problem as: Find* $(\tilde{c}, \tilde{p}) \in \mathcal{K}^1_{\vec{a}+1}(\Omega) \times \mathcal{K}^{1,0}_{\vec{a}+1}(\Omega)$ *such that*

$$\begin{aligned} \Phi(\tilde{p};c_h) &= q^I - q^P, \\ \Psi(\tilde{c};p_h) &= \hat{c}q^I \end{aligned} \tag{6.6.12}$$

where (c_h, p_h) *is the solution to the discrete problem.*

We first consider Assumption 6.1.5 in the weighted spaces. We have $\mathcal{K}^1_{\vec{a}+1}(\Omega) \subset H^1(\Omega)$ for $\vec{a} \geq 0$. Therefore we may apply the Scott-Zhang interpolation operator as we did in Lemma 6.2.22, using now Assumption 6.6.4.

Lemma 6.6.13. *Let* $(\tilde{c}, \tilde{p}) \in \mathcal{K}^1_{\vec{a}+1}(\Omega) \times \mathcal{K}^{1,0}_{\vec{a}+1}(\Omega)$ *be the solution of* (6.6.12)*, and let* $(c_h, p_h) \in V_{cG} \times (V_{cG} \cap L^2_0(\Omega))$ *be the approximation defined in* (6.6.10) *and assume Assumptions 6.6.1 and 6.6.4 hold. With* $q^I = q^I_h$, $q^P = q^P_h$ *and* $\hat{c} = \hat{c}_h$ *we have the following a posteriori error estimators:*

$$\begin{aligned} \|\tilde{p} - p_h\|^2_{\mathcal{K}^1_{\vec{a}+1}(\Omega)} \leq C_p \Bigg(& \sum_{E \in \mathcal{T}_h} h_E^2 \|q^I - q^P + \nabla \cdot (a(c_h)\nabla p_h)\|^2_{L^2(E)} \\ & + \sum_{e \in \mathcal{E}^o_h} h_e \|[\![a(c_h)\nabla p_h]\!]\|^2_{L^2(e)} \Bigg) \end{aligned} \tag{6.6.14}$$

and

$$\begin{aligned}&\|\tilde{c}-c_h\|^2_{\mathcal{K}^1_{\vec{a}+1}(\Omega)}\\ &\leq C_c\left(\sum_{E\in\mathcal{T}_h} h_E^2\|\mathtt{R_c}\|^2_{L^2(E)}+\sum_{e\in\mathcal{E}_h^o} h_e\|[\![\mathbb{D}(u_h)\nabla c_h]\!]+\frac{1}{2}[\![u_h c_h]\!]\|^2_{L^2(e)}\right)\end{aligned}\tag{6.6.15}$$

where $\mathtt{R_c}$ *is as defined in* (6.2.25)*. Here* C_p *depends on the coercivity coefficient in* (6.6.5) *and* C_{sz} *in* (3.3.3)*, and* C_c *depends on the coercivity coefficient in* (6.6.6) *and* C_{sz}*.*

Proof. Follow the steps of Lemma 6.2.22, using instead the coercivity given in Assumption 6.6.4 and the orthogonality generated from the terms of Definitions 6.6.9 and 6.6.11. □

To show the coupling assumption in the weighted spaces is more difficult. As we have already remarked we do not have a result concerning weighted boundedness of ∇p. We cannot treat the term $((\mathbb{D}(u_h)-\mathbb{D}(u))\nabla c, \nabla(c-\tilde{c}))$ while keeping all terms normed in $\mathcal{K}^1_{\vec{a}+1}(\Omega)$ (note that the term contains derivatives on all three parts through the definition of u). We therefore have to reduce the norm imposed on c. This leads to the unexpected appearance of $\mathcal{K}^0_{1-\vec{a}}(\Omega)$ control on c.

Lemma 6.6.16. *Let* (c,p) *and* $(\tilde{c},\tilde{p})\in\mathcal{K}^1_{\vec{a}+1}(\Omega)\times\mathcal{K}^{1,0}_{\vec{a}+1}(\Omega)$ *be the solution to the continuous problem defined in Definition 6.6.7 and the auxiliary problem defined in Definition 6.6.11 respectively. Assume Assumptions 6.6.1 and 6.6.4 hold. Then for* $1/3\leq\vec{a}\leq 1$ *and real* $2<q<\infty$ *we have*

$$\|p-\tilde{p}\|^2_{\mathcal{K}^1_{\vec{a}+1}(\Omega)}\leq\gamma_p\|c-c_h\|^2_{\mathcal{K}^0_{1-\vec{a}}(\Omega)}\tag{6.6.17}$$

where

$$\gamma_p=\left(\frac{a^\circ\|\vartheta^{1-\vec{a}}\nabla p\|_{L^q(\Omega)}}{\Xi_p}\right)^2,\tag{6.6.18}$$

and

$$\|c-\tilde{c}\|^2_{\mathcal{K}^0_{1-\vec{a}}(\Omega)}\leq\gamma_c\|p-p_h\|^2_{\mathcal{K}^1_{\vec{a}+1}(\Omega)}\tag{6.6.19}$$

where

$$\gamma_c = \left(C_{BK} a^\circ \left(d^\circ \|\vartheta^{2\vec{a}} \nabla \tilde{c}\|_{L^q(\Omega)} + \frac{1}{2} \|\vartheta^{2\vec{a}+1} \nabla \tilde{c}\|_{L^q(\Omega)} + \frac{1}{2} \|\vartheta^{2\vec{a}} \tilde{c}\|_{L^\infty(\Omega)} \right) \right)^2 \tag{6.6.20}$$

where C_{BK} is as defined in Theorem 6.4.11.

Proof. The proof for Φ closely follows that from Lemma 6.2.16. By subtracting the Φ terms in (6.6.8) from those in (6.6.12) we find $\Phi(\tilde{p}; c_h) - \Phi(p; c) = 0$. Then using the coercivity of Φ from Assumption 6.6.4 we have

$$\begin{aligned}
\Xi_p \|p - \tilde{p}\|^2_{\mathcal{K}^1_{\vec{a}+1}(\Omega)} &\leq \Phi(p - \tilde{p}; c_h)[p - \tilde{p}] \\
&= \Phi(p; c_h)[p - \tilde{p}] - \Phi(p; c)[p - \tilde{p}] - \Phi(\tilde{p}; c_h)[p - \tilde{p}] + \Phi(p; c)[p - \tilde{p}] \\
&= (\vartheta^{2\vec{a}-1}(a(c_h) - a(c))\vartheta^{1-\vec{a}} \nabla p, \vartheta^{-\vec{a}} \nabla (p - \tilde{p})) \\
&\leq \|\vartheta^{2\vec{a}-1}(a(c_h) - a(c))\|_{L^{2+\delta}(\Omega)} \|\vartheta^{1-\vec{a}} \nabla p\|_{L^q(\Omega)} \|\vartheta^{-\vec{a}} \nabla (p - \tilde{p})\|_{L^2(\Omega)} \\
&\leq a^\circ \|c - c_h\|_{\mathcal{K}^0_{1-\vec{a}}(\Omega)} \|\vartheta^{1-\vec{a}} \nabla p\|_{L^q(\Omega)} \|p - \tilde{p}\|_{\mathcal{K}^1_{\vec{a}+1}(\Omega)}
\end{aligned}$$

where we have used the Sobolev embedding theorems in the weighted spaces of Section 6.4, chosen $2 < q < \infty$ such that $1/2 + 1/(2+\delta) + 1/q = 1$ and used $\vartheta^{2\vec{a}-1}(c - c_h) \leq \vartheta^{\vec{a}-1}(c - c_h)$. With rearrangement we have shown (6.6.17).

For concentration our approach is similar to that in Chapter 5. First introduce the dual equation on the domain Ω.

$$\begin{aligned}
\nabla \cdot (\mathbb{D}(u) \nabla \zeta) + \frac{1}{2} u \cdot \nabla \zeta + \frac{1}{2} \nabla \cdot (u\zeta) - \frac{1}{2}(q^I + q^P)\zeta = \vartheta^b (c - \tilde{c}) \quad &\text{on } \Omega, \\
(\mathbb{D}(u) \nabla \zeta) \cdot n = 0 \quad &\text{on } \partial\Omega,
\end{aligned} \tag{6.6.21}$$

where b is a real valued vector at our disposal. Using Assumption 6.6.1 we have that $\zeta \in \mathcal{K}^{2,0}_{\vec{a}+1}(\Omega)$ for the same $\vec{a}$ as in Definition 6.6.7 and we have the regularity result

$$\|\zeta\|_{\mathcal{K}^2_{\vec{a}+1}(\Omega)} \leq C_{BK} \|\vartheta^b (c - \tilde{c})\|_{\mathcal{K}^0_{\vec{a}-1}(\Omega)} = C_{BK} \|c - \tilde{c}\|_{\mathcal{K}^0_{\vec{a}-1-b}(\Omega)} \tag{6.6.22}$$

where $C_{BK} > 0$ is a constant independent of c and $\tilde{c}$. Using (6.6.21), the regularity

of u and ζ and integrating we find

$$\begin{aligned}\|c-\tilde{c}\|^2_{\mathcal{K}^0_{-b/2}(\Omega)} &= (\vartheta^{b/2}(c-\tilde{c}),\vartheta^{b/2}(c-\tilde{c}))\\ &= (\nabla\cdot(\mathbb{D}(u)\nabla\zeta),c-\tilde{c})+\frac{1}{2}(u\cdot\nabla\zeta,c-\tilde{c})\\ &\quad+\frac{1}{2}(\nabla\cdot(u\zeta),c-\tilde{c})-\frac{1}{2}((q^I+q^P)\zeta,c-\tilde{c})\\ &= -(\mathbb{D}(u)\nabla\zeta,\nabla(c-\tilde{c}))+\frac{1}{2}(u\cdot\nabla\zeta,c-\tilde{c})\\ &\quad-\frac{1}{2}(u\zeta,\nabla(c-\tilde{c}))-\frac{1}{2}((q^I+q^P)\zeta,c-\tilde{c}).\end{aligned}$$

Subtracting Ψ terms in Definition 6.6.7 and 6.6.11 and using the test function ζ in the weak form gives $\Psi(c;p)[\zeta]-\Psi(\tilde{c};p_h)[\zeta]=0$. By adding this to the previous equation we find

$$\begin{aligned}&\|c-\tilde{c}\|^2_{\mathcal{K}^0_{-b/2}(\Omega)}\\ &= (\mathbb{D}(u)\nabla c-\mathbb{D}(u_h)\nabla\tilde{c},\nabla\zeta)-(\mathbb{D}(u)\nabla\zeta,\nabla(c-\tilde{c}))\\ &\quad+\frac{1}{2}(u\cdot\nabla c-u_h\cdot\nabla\tilde{c},\zeta)-\frac{1}{2}(u\zeta,\nabla(c-\tilde{c}))\\ &\quad-\frac{1}{2}(uc-u_h\tilde{c},\nabla\zeta)+\frac{1}{2}(u\cdot\nabla\zeta,c-\tilde{c})\\ &= ((\mathbb{D}(u)-\mathbb{D}(u_h))\nabla\tilde{c},\nabla\zeta)+\frac{1}{2}((u-u_h)\nabla\tilde{c},\zeta)-\frac{1}{2}((u-u_h)\tilde{c},\nabla\zeta).\end{aligned}$$

For the first term we find

$$\begin{aligned}((\mathbb{D}(u)-\mathbb{D}(u_h))\nabla\tilde{c},\nabla\zeta) &\le d^\circ a^\circ(\vartheta^{-\vec{a}}\nabla(p-\tilde{p})\vartheta^{2\vec{a}}\nabla\tilde{c},\vartheta^{-\vec{a}}\nabla\zeta)\\ &\le d^\circ a^\circ\|\vartheta^{-\vec{a}}\nabla(p-p_h)\|_{L^{2+\delta}(\Omega)}\|\vartheta^{2\vec{a}}\nabla\tilde{c}\|_{L^q(\Omega)}\|\vartheta^{-\vec{a}}\nabla\zeta\|_{L^2(\Omega)}\\ &\le d^\circ a^\circ\|p-p_h\|_{\mathcal{K}^1_{\vec{a}+1}(\Omega)}\|\vartheta^{2\vec{a}}\nabla\tilde{c}\|_{L^q(\Omega)}\|\zeta\|_{\mathcal{K}^2_{\vec{a}+1}(\Omega)},\end{aligned}$$

where q and δ are chosen as above. Now we may apply Lemma 6.5.8 to the third term provided $2\vec{a}\ge 1-\vec{a}$ which holds as $\vec{a}\ge 1/3$. We repeat this process for the

other terms, then using (6.6.22) we find

$$
\begin{aligned}
&\|c-\tilde{c}\|^2_{\mathcal{K}^0_{-b/2}(\Omega)} \\
&\leq C_{BK}a^\circ\left(d^\circ\|\vartheta^{2\vec{a}}\nabla\tilde{c}\|_{L^q(\Omega)}+\frac{1}{2}\|\vartheta^{2\vec{a}+1}\nabla\tilde{c}\|_{L^q(\Omega)}\right. \\
&\qquad\left.+\frac{1}{2}\|\vartheta^{2\vec{a}}\tilde{c}\|_{L^\infty(\Omega)}\right)\|p-p_h\|_{\mathcal{K}^1_{\vec{a}+1}(\Omega)}\|c-\tilde{c}\|_{\mathcal{K}^0_{\vec{a}-1-b}(\Omega)}.
\end{aligned}
\tag{6.6.23}
$$

The optimum choice for b is when $-b/2=\vec{a}-1-b\Rightarrow b=2\vec{a}-2$ which gives (6.6.19). □

Assumption 6.6.24. *We assume that γ_c and γ_p defined in Lemma 6.6.16 satisfy $\gamma_c\gamma_p<1$.*

We may now combine the results (6.6.14), (6.6.15), (6.6.17) and (6.6.19) in the manner of Section 6.1.

Theorem 6.6.25. *Assume that Assumptions 6.6.1, 6.6.4 and 6.6.24 hold, and that the conditions of Lemma 6.6.13 and 6.6.16 hold. Then the solution $(c,p)\in\mathcal{K}^2_{\vec{a}+1}(\Omega)\times\mathcal{K}^{2,0}_{\vec{a}+1}(\Omega)$ to (6.6.8) and $(p_h,c_h)\in V_{cG}\times(V_{cG}\cap L^2_0(\Omega))$ defined by (6.6.10) satisfy the a posteriori error estimate*

$$
\begin{aligned}
&\|p-p_h\|^2_{\mathcal{K}^1_{\vec{a}+1}(\Omega)}+\|c-c_h\|^2_{\mathcal{K}^0_{1-\vec{a}}(\Omega)} \\
&\leq C\left(\sum_{E\in\mathcal{T}_h}h_E^2\left(\|q^I-q^P+\nabla\cdot(a(c_h)\nabla p_h)\|^2_{L^2(E)}+\|\mathtt{R_c}\|^2_{L^2(E)}\right)\right. \\
&\qquad\left.+\sum_{e\in\mathcal{E}_h^o}h_e\left(\|[\![a(c_h)\nabla p_h]\!]\|^2_{L^2(e)}+\|[\![\mathbb{D}(u_h)\nabla c_h]\!]+\frac{1}{2}[\![u_hc_h]\!]\|^2_{L^2(e)}\right)\right)
\end{aligned}
\tag{6.6.26}
$$

where C depends on the constants defined in Assumption 6.6.4, 6.6.13 and 6.6.16.

Proof. Following the steps of the abstract proof of Theorem 6.1.8 we combine the results of Lemmas 6.6.16 and 6.6.13. Note that the estimator (6.6.15) automatically provides control on $\mathcal{K}^0_{1-\vec{a}}(\Omega)$. □

Remark 6.6.27. *This bound only offers control on $c-c_h$ in the $\mathcal{K}^0_{1-\vec{a}}(\Omega)$ norm for the reasons discussed. If we make the additional assumption that we can bound the*

derivatives, e.g., $\vartheta^{1-\vec{a}}\nabla p \in L^\infty(\Omega)$ *we can achieve the same control on both terms in the manner of Section 6.2.*

6.7 A Review of Our Error Estimators

We conclude this part with a brief review of the error estimators we have presented.

In Chapter 5 we formulated an a posteriori error estimator for the coupled, time dependent problem of incompressible miscible displacement. In order to do so we assumed that the derivatives of pressure and concentration were bounded, the domain Ω was convex and that the coefficients which were derived during the analysis satisfied a particular relationship (Assumption 5.4.1).

In the current chapter we considered first the fundamental aspects of the analysis. We identified several properties which, when possessed by a coupled system of equations and their approximation, lead to an a posteriori error estimator for the coupled problem. This abstract approach is applicable to stationary and time dependent problems. Note however that the analysis of Chapter 5 does not follow the scheme exactly. The estimator for the concentration in Theorem 5.3.6 is constructed using the dual equation (5.3.1)-(5.3.3), not with an auxiliary equation in the manner of Section 6.1, and the control on $\mathtt{E_c}$ is in L^2 not H^1.

We derive an a posteriori error estimator for the stationary problem of incompressible miscible displacement in Section 6.2, making the same assumptions on the boundedness of the derivatives as in Chapter 5. We note however that the assumption of boundedness is not attained for some domains or for some conditions on the assumptions. This motivates the study of weighted spaces. The estimator that we derive for the weighted case must be constructed without relying on the boundedness of the derivatives and with only modified (i.e., weighted) control on the boundedness of the solution. If it were possible to show that the derivatives were bounded after application of some weight we could generate the same control on both $p - p_h$ and $c - c_h$. As it is we have to formulate the analysis to give control only in $\mathcal{K}^0_{1-\vec{a}}(\Omega)$ on $c - c_h$.

To compare the estimators (with reservation) we note that, e.g., $\|p-p_h\|_{\mathcal{K}^1_{\vec{a}+1}(\Omega)} \geq$

$\|p - p_h\|_{H^1(\Omega)}$ (provided of course that p has sufficient regularity). Therefore, depending on the values of the constants γ_p and γ_c for the weighted and unweighted problem, we see that the weighted bound may be more reliable (in the sense that the right hand side of (6.2.29) is a more distant predictor of error than the right hand side of (6.6.26)). This suggests that there is some scope to introduce "artificial vertices", i.e., vertices with no associated singularity, to achieve more reliable a posteriori bounds.

Part III

Constraining the Jumps in the Discontinuous Galerkin Method

Chapter 7

Continuous-Discontinuous Galerkin Methods by Local Super Penalization

The material included in this chapter has been published in part in [46].

We now return to discussion of the continuous discontinuous Galerkin method. The control of discontinuities across element interfaces in the dG framework can be exercised by introducing and/or tuning the, so-called, jump penalization parameters (that is σ in (2.2.3) and (4.4.6)). Using excessive penalization within a dG approximation will be referred to as the *super penalty method*. It is natural to expect that as the penalty parameter is increased the interelement jumps in the numerical approximation decrease. It has been shown by Larson and Niklasson [96] for stationary linear elliptic problems (using the interior penalty method) and by Burman, Quarteroni and Stamm [42] for stationary hyperbolic problems (penalizing the jumps of the approximation for discontinuous elements and the jumps in the gradient of the approximation for continuous elements) that the dG approximation converges to the cG approximation as the jump penalization parameter tends to infinity.

Firstly, we present an alternative proof of the convergence of dG methods to cG methods, using a far more general framework covering the cases considered by [42, 96] and also non-linear and time dependent problems. Moreover, we show that super penalization procedures can be localized to designated element faces, thereby

arriving at the cdG method as described in Chapter 2. As particular examples we consider the limits of the interior penalty dG method for PDEs with non-negative characteristic form [82] and the mixed Raviart-Thomas-dG method for the miscible displacement system presented in Chapter 4.

7.1 An Abstract Discussion

Consider a (possibly non-linear) operator $\mathcal{B} : W \times W \to \mathbb{R}$ where W is a finite dimensional vector space with norm $\|\cdot\|_W$. Suppose there exists a decomposition of W such that $V \oplus X = W$ for $V, X \subset W$. In particular this means we can write any $w \in W$ uniquely as $w = v + x$ for some $v \in V$ and $x \in X$.

Assume that $\mathcal{B}$ is coercive, i.e., there exists $\Lambda_W > 0$ (typically independent of the dimension of W), such that

$$\mathcal{B}(w, w) \geq \Lambda_W \|w\|_W^2 \qquad \forall\ w \in W. \tag{7.1.1}$$

Consider another operator $\mathcal{S} : W \times W \to \mathbb{R}$, whose support is restricted to $X \times X$ in the sense that

$$\mathcal{S}(v, \hat{v}) = 0 \qquad \forall\ v, \hat{v} \in V \tag{7.1.2}$$

and

$$\mathcal{S}(v, x) = \mathcal{S}(x, v) = 0 \qquad \forall\ v \in V,\ x \in X. \tag{7.1.3}$$

We require coercivity on X, i.e., there exists $\Lambda_X > 0$ such that for all $x \in X$

$$\mathcal{S}(x, x) \geq \Lambda_X \|x\|_X^2, \tag{7.1.4}$$

where $\|x\|_X$ is a norm on X. In view of (7.1.2) this gives $\mathcal{S}(w, w) \geq \Lambda_X \|w\|_X^2 = \Lambda_X \|x\|_X^2$. We construct a further operator

$$\mathcal{B}_\sigma := \mathcal{B} + \sigma \mathcal{S} \tag{7.1.5}$$

where $0 \leq \sigma \in \mathbb{R}$, and call this the *super penalized* bilinear form.

Let ℓ be an element of the dual space W^* of W, independent of σ. Then choose $w_\sigma \in W$ such that

$$\mathcal{B}_\sigma(w_\sigma, w) = \ell(w) \qquad \forall\ w \in W. \tag{7.1.6}$$

Also choose $v_h \in V$ such that

$$\mathcal{B}(v_h, v) = \ell(v) \qquad \forall\ v \in V. \tag{7.1.7}$$

Observe that for all $\sigma \in \mathbb{R}$

$$\mathcal{B}_\sigma(v_h, v) = \mathcal{B}(v_h, v) = \ell(v) \qquad \forall\ v \in V \tag{7.1.8}$$

using (7.1.2). Now with (7.1.1), (7.1.4) and (7.1.6) we have

$$\begin{aligned} \Lambda_W \|w_\sigma\|_W^2 + \sigma \Lambda_X \|w_\sigma\|_X^2 &\leq \mathcal{B}(w_\sigma, w_\sigma) + \sigma \mathcal{S}(w_\sigma, w_\sigma) \\ &= \mathcal{B}_\sigma(w_\sigma, w_\sigma) \\ &= \ell(w_\sigma) \\ &\leq \|\ell\|_{W^*} \|w_\sigma\|_W. \end{aligned}$$

Using Young's inequality we see

$$\frac{\Lambda_W^2}{\sigma} \|w_\sigma\|_W^2 + 2\Lambda_W \Lambda_X \|w_\sigma\|_X^2 \leq \frac{1}{\sigma} \|\ell\|_{W^*}^2. \tag{7.1.9}$$

Each of Λ_W, Λ_X and $\|\ell\|_{W^*}$ are independent of σ. We write $w_\sigma = v_\sigma + x_\sigma$, the unique decomposition with $v_\sigma \in V$ and $x_\sigma \in X$. From (7.1.9) we see

$$\lim_{\sigma \to \infty} \|v_\sigma + x_\sigma\|_X = \lim_{\sigma \to \infty} \|x_\sigma\|_X = 0. \tag{7.1.10}$$

Therefore $x_\sigma \to 0$ as $\sigma \to \infty$.

Now assume that $\mathcal{B}$ is continuous in the first argument in the following sense: If

$\lim_{i\to\infty} w_i = w \in W$ then

$$\lim_{i\to\infty} \mathcal{B}(w_i, v) = \mathcal{B}(w, v) \qquad \forall\ v \in V. \tag{7.1.11}$$

Suppose $w_\sigma \nrightarrow v_h$ as $\sigma \to \infty$. Then there exists $\varepsilon > 0$ such that there is some sequence $\{w_{\sigma(i)}\}_i$ with $\sigma(i) \to \infty$ as $i \to \infty$ satisfying

$$\|w_{\sigma(i)} - v_h\|_W > \varepsilon \qquad \forall i \in \mathbb{N}. \tag{7.1.12}$$

Owing to (7.1.9) the sequence $\{w_{\sigma(i)}\}_i$ is a bounded subset of W. Then by the Heine-Borel Theorem there exists a convergent subsequence, also denoted $\{w_{\sigma(i)}\}_i$, such that

$$\tilde{w} = \lim_{i\to\infty} w_{\sigma(i)}. \tag{7.1.13}$$

Considering (7.1.10) we know that $\tilde{w} \in V$. We have that for all $v \in V$

$$\begin{aligned}
\mathcal{B}(\tilde{w}, v) &= \mathcal{B}\left(\lim_{i\to\infty} w_{\sigma(i)}, v\right) \\
&= \lim_{i\to\infty} \mathcal{B}(w_{\sigma(i)}, v) && \text{by (7.1.11)} \\
&= \lim_{i\to\infty} \mathcal{B}_\sigma(w_{\sigma(i)}, v) && \text{by (7.1.3)} \\
&= \lim_{i\to\infty} \ell(v) && \text{by (7.1.6)} \\
&= \ell(v).
\end{aligned}$$

Hence $\tilde{w}$ satisfies (7.1.7) and by (7.1.13) we have

$$\lim_{i\to\infty} \|w_{\sigma(i)} - v_h\|_W = 0.$$

This contradicts (7.1.12) and we conclude that all subsequences $\{w_{\sigma(i)}\}_i$ converge to v_h. Therefore

$$\lim_{\sigma\to\infty} (w_\sigma - v_h) = 0. \tag{7.1.14}$$

We finally remark on the potential loss of stability due to super penalization. It can be seen from (7.1.9) that as $x_\sigma \to 0$ when $\sigma \to \infty$ the coercivity of $\mathcal{B}_\sigma$ is increasingly compromised, which can lead to loss of stability and reduction on the rate of convergence in various settings.

7.2 Equations of Non-Negative Characteristic Form

In this chapter we examine a more general linear equation than that discussed in Chapter 2, namely [82]

$$\begin{aligned} -\nabla \cdot (\mathbb{A}(\boldsymbol{x})\nabla u) + b(\boldsymbol{x}) \cdot \nabla u + c(\boldsymbol{x})u = f(\boldsymbol{x}) \qquad & \text{in } \Omega, \\ u = 0 \qquad & \text{on } \partial\Omega \end{aligned} \tag{7.2.1}$$

with b a $\mathbb{R}^d$ valued function whose entries are Lipschitz continuous on $\overline{\Omega}$, $c \in L^\infty(\Omega)$ and $f \in L^2(\Omega)$ real valued functions. The diffusion coefficient $\mathbb{A}$ is a $d \times d$ symmetric matrix with entries being bounded, piecewise continuous real-valued functions defined on $\overline{\Omega}$, with

$$\zeta^\top \mathbb{A} \zeta \geq 0 \qquad \forall \zeta \in \mathbb{R}^d, \text{ a.e. } \boldsymbol{x} \in \overline{\Omega}.$$

With these conditions (7.2.1) is named a *partial differential equation with non-negative characteristic form*. We consider this more general equation so that we include the hyperbolic case (i.e., $\mathbb{A} = 0$) and cases which may not be singularly perturbed.

In the notation of Section 7.1 we identify $W = V_{\text{dG}}$, $V = V_{\text{cdG}}$ and define $V_{\text{dG}} := V_{\text{cdG}} \oplus V_\perp$, so $X = V_\perp$, the space of piecewise polynomials strictly discontinuous on $\mathcal{T}_{\text{cG}}$ and matching V_{dG} on $\mathcal{T}_{\text{dG}}$. Note that the standard continuous space is obtained by setting $\mathcal{T}_h = \mathcal{T}_{\text{cG}}$. When there is no diffusion term we adjust the cdG space so that the boundary conditions are only imposed on the inflow boundary, i.e.,

$$V_{\text{cdG}} := \{v \in L^2(\Omega) : \forall E \in \mathcal{T}_h, v|_E \in \mathbb{P}^k, v|_{\Gamma_{\text{cG}} \cap \Gamma^{\text{in}}} = 0, v|_{\mathcal{T}_{\text{cG}}} \in C(\overline{\mathcal{T}}_{\text{cG}})\}.$$

Define $\mathcal{B} : V_{\mathrm{dG}} \times V_{\mathrm{dG}} \to \mathbb{R}$, the bilinear form for the interior penalty family of methods with $\vartheta \in \{-1, 1\}$ for (7.2.1), by

$$\mathcal{B}(w, \hat{w}) := \mathcal{B}_d(w, \hat{w}) + \mathcal{B}_{ar}(w, \hat{w}) \tag{7.2.2}$$

with

$$\begin{aligned} \mathcal{B}_d(w, \hat{w}) := &\sum_{E \in \mathcal{T}_h} \int_E \mathbb{A} \nabla_h w \cdot \nabla_h \hat{w} \, \mathrm{d}\boldsymbol{x} + \sum_{e \in \mathcal{E}_h} \int_e m [\![w]\!] \cdot [\![\hat{w}]\!] \, \mathrm{d}s \\ &- \sum_{e \in \mathcal{E}_h} \int_e \Big(\{\!\{ \mathbb{A} \nabla_h w \}\!\} \cdot [\![\hat{w}]\!] - \vartheta \{\!\{ \mathbb{A} \nabla_h \hat{w} \}\!\} \cdot [\![w]\!] \Big) \, \mathrm{d}s \end{aligned} \tag{7.2.3}$$

and

$$\begin{aligned} \mathcal{B}_{ar}(w, \hat{w}) := &\sum_{E \in \mathcal{T}_h} \int_E (b \cdot \nabla_h w) \hat{w} + c w \hat{w} \, \mathrm{d}\boldsymbol{x} \\ &- \sum_{e \in \mathcal{E}_h^o} \int_e b \cdot [\![w]\!] \hat{w}^{\mathrm{out}} \, \mathrm{d}s - \sum_{e \in \Gamma^{\mathrm{in}}} \int_e (b \cdot n) w \hat{w} \, \mathrm{d}s. \end{aligned} \tag{7.2.4}$$

This is reduces to (2.2.6) when $\mathbb{A} = \varepsilon \mathbb{I}$ and also adjusting notation on the penalty term. We define $m := C_p \{\!\{ \overline{\mathbb{A}} r^2 \}\!\} / h_e$, $\overline{\mathbb{A}} := \| |\sqrt{\mathbb{A}}|_2 \|_{L^\infty(E)}$, with $|\cdot|_2$ denoting the matrix-2-norm, and $C_p(\vartheta) \geq 0$ fixed for a given ϑ. The linear form is given by

$$\ell(w) := \sum_{E \in \mathcal{T}_h} \int_E f w \, \mathrm{d}\boldsymbol{x}. \tag{7.2.5}$$

For $e \in \mathcal{E}_{\mathrm{cG}}$ we have the additional term $\mathcal{S} : V_{\mathrm{dG}} \times V_{\mathrm{dG}} \to \mathbb{R}$ penalizing the jumps where

$$\mathcal{S}(w, \hat{w}) := \sum_{e \in \mathcal{E}_{\mathrm{cG}}} \int_e M [\![w]\!] \cdot [\![\hat{w}]\!] \, \mathrm{d}s \tag{7.2.6}$$

and

$$M := \left(C_{ar} + C_d \frac{\{\!\{ \overline{\mathbb{A}} r^2 \}\!\}}{h_e} \right)$$

with C_{ar} and C_d fixed constants independent of σ. Then we define $\mathcal{B}_\sigma(w, \hat{w}) := \mathcal{B}(w, \hat{w}) + \sigma \mathcal{S}(w, \hat{w})$. Notice that we have two penalty type terms, m and M. In

this way we can fix C_p large enough to ensure coercivity independently of $\sigma \to \infty$.

Remark 7.2.7. *Choosing $\mathbb{A} = \varepsilon\mathbb{I}$, where $\mathbb{I}$ is the $d \times d$ identity matrix, returns the singularly perturbed advection diffusion reaction equation (1.1.1). Observe that if we take $C_{ar} = C_d = 0$ (or $\sigma = 0$) we recover the usual interior penalty method. If we take $C_p = C_d = C_{ar} = 0$ and $\mathbb{A} = 0$ we have the standard (unpenalized) bilinear form for the purely hyperbolic equation (assuming of course that we adjust the boundary conditions appropriately). Taking $C_d = 0$ and $C_{ar} \neq 0$ when $\mathbb{A} = 0$ gives the method proposed in [37], i.e., a method penalizing only the jumps in the solution, but not the jumps in the gradient, cf., [42].*

All functions in V_{cdG} are continuous on edges in $\mathcal{E}_{\text{cG}}$ (recall that by definition edges in J are not included in $\mathcal{E}_{\text{cG}}$). Therefore conditions (7.1.2) and (7.1.3) are satisfied for this $\mathcal{S}$. That is, for any $v, \hat{v} \in V_{\text{cdG}}$ and $x \in V_\perp$

$$\mathcal{S}(v, \hat{v}) = \mathcal{S}(v, x) = \mathcal{S}(x, v) = 0. \tag{7.2.8}$$

We define the following norm for all $w \in V_{\text{dG}}$.

$$\begin{aligned} |||w|||^2 := &\sum_{E\in\mathcal{T}_h} \|\sqrt{\mathbb{A}}\nabla_h w\|^2_{L^2(E)} + \|r^{1/2}w\|^2_{L^2(\Omega)} \\ &+ \sum_{e\in\mathcal{E}_h} \frac{1}{2}\||b\cdot n|^{1/2}[\![w]\!]\|^2_{L^2(e)} + \sum_{e\in\mathcal{E}_h} \|\sqrt{m}[\![w]\!]\|^2_{L^2(e)} \end{aligned} \tag{7.2.9}$$

where $r := c - {}^1\!/_2\nabla_h \cdot b$. We also define for $w \in V_{\text{dG}}$

$$|w|^2_{\mathcal{S}} := \sum_{e\in\mathcal{E}_{\text{cG}}} \|\sqrt{M}[\![w]\!]\|^2_{L^2(e)}. \tag{7.2.10}$$

Notice that $|\cdot|_{\mathcal{S}}$ is a semi-norm on V_{dG} but a norm on $V_\perp$. To make this distinction clear we will write $\|x\|_{\mathcal{S}}$ for $x \in V_\perp$.

Lemma 7.2.11. *If C_p is sufficiently large when $\vartheta = -1$ then $\mathcal{B}$ is coercive on V_{dG}, i.e., for all $w \in V_{dG}$*

$$\mathcal{B}(w, w) \geq \Lambda_{cc}|||w|||^2 \tag{7.2.12}$$

with $\Lambda_W = 1$ *when* $\vartheta = 1$ *and* $\Lambda_W = 1/2$ *when* $\vartheta = -1$.

Proof. We will consider the coercivity in more detail in Chapter 8 for the interior penalty method for (1.1.1). Given the definition of the norm (7.2.9) compared to (2.2.9) we see that the proof for equations of non-negative characteristic form will proceed in the same way. □

From the definition it is clear that $\mathcal{S}$ is coercive with constant one on $V_\perp$, i.e., for all $x \in V_\perp$

$$\mathcal{S}(x,x) = \|x\|_{\mathcal{S}}^2. \tag{7.2.13}$$

Definition 7.2.14. *Define a dG approximation to* (7.2.1) *as* $w_\sigma \in V_{dG}$ *satisfying*

$$\mathcal{B}_\sigma(w_\sigma, w) = \ell(w) \qquad \forall w \in V_{dG}. \tag{7.2.15}$$

Definition 7.2.16. *Define a cdG approximation to* (7.2.1) *as* $v_h \in V_{dG}$ *satisfying*

$$\mathcal{B}_\sigma(v_h, v) = \ell(v) \qquad \forall v \in V_{cdG}. \tag{7.2.17}$$

Using (7.2.8) we see that v_h also satisfies $\mathcal{B}(v_h, v) = \ell(v)$ for all $v \in V_{\text{cdG}}$.

Theorem 7.2.18. *The dG finite element approximation* w_σ *converges to the cdG finite element approximation* v_h *as* $\sigma \to \infty$, *i.e.,*

$$\lim_{\sigma\to\infty} (w_\sigma - v_h) = 0.$$

Proof. Following the argument of Section 7.1 we use Lemma 7.2.11 and (7.2.13) and note that (7.1.11) is satisfied as linear operators in finite-dimensional vector spaces are continuous. □

7.3 Equations of Incompressible Miscible Displacement

Recall that in Chapter 4 we introduced the continuous time RT-dG finite element method where we solved for the pressure and velocity using a Raviart-Thomas (RT) procedure and for the concentration using a dG method. We now consider the discrete time RT-cdG finite element method, solving for pressure and velocity as before but modifying the approximation scheme for concentration. As we now use discrete time we use the notation that, e.g., c_h^j now refers to the approximation of concentration at timestep j.

As previously the velocity and pressure are approximated in $U \times P$ defined in Chapter 4. To simplify the presentation we use the same mesh $\mathcal{T}_h$ to solve for u, p and c numerically at each time step and there is no refinement of the mesh or polynomial degree. However $\mathcal{T}_{\mathrm{cG}}$ and $\mathcal{T}_{\mathrm{dG}}$ are not fixed so the cdG space used to approximate c will vary with time. We define the time dependent cdG space by

$$V_{\mathrm{cdG}}^j := \{v \in L^2(\Omega) : \forall E \in \mathcal{T}_h, v|_E \in \mathbb{P}^k, v|_{\Gamma_{\mathrm{cG}}^j} = 0, v|_{\mathcal{T}_{\mathrm{cG}}^j} \in C(\overline{\mathcal{T}}_{\mathrm{cG}}^j)\}$$

where $\mathcal{T}_{\mathrm{cG}}^j$ and Γ_{cG}^j are the $\mathcal{T}_{\mathrm{cG}}$ region and external boundary of $\mathcal{T}_{\mathrm{cG}}^j$ at time t_j. As we assert that no change to the shape of the mesh occurs in time we define the time dependent dG space as in (2.2.2). Then we may define $V_\perp^j$ via $V_{\mathrm{dG}}^j = V_{\mathrm{cdG}}^j \oplus V_\perp^j$. Note that the degree l is the same for U and P but need not be equal to k, the degree of the polynomials used to approximate concentration.

Let $0 = t_0 < t_1 < \ldots < t_N = T$ be a partition of the time interval $[0, T]$. For simplicity we assume that each time step is of equal length and define $\Delta t := t_j - t_{j-1}$ and the backward Euler operator $\mathfrak{d}_{\mathrm{t}} c_h^j := (\Delta t)^{-1}(c_h^j - c_h^{j-1})$ for $j = 1, 2, \ldots, N$. To keep our notation consistent with the abstract analysis in Section 7.1 we define (cf.,

(4.4.6) where we used σ for the penalty parameter)

$$\begin{aligned}\mathcal{B}_d(c_h^j, d_h^j; u_h^j) = &\sum_{E\in\mathcal{T}_h}\int_E \mathbb{D}(u_h^j)\nabla_h c_h^j \cdot \nabla_h d_h^j \,\mathrm{d}\boldsymbol{x} + \sum_{e\in\mathcal{E}_h^o}\int_e m[\![c_h^j]\!]\cdot[\![d_h^j]\!]\,\mathrm{d}s \\ &- \sum_{e\in\mathcal{E}_h^o}\int_e [\![c_h^j]\!]\cdot\{\!\{\mathbb{D}(u_h^j)\nabla_h d_h^j\}\!\} + [\![d_h^j]\!]\cdot\{\!\{\mathbb{D}(u_h^j)\nabla_h c_h^j\}\!\}\,\mathrm{d}s\end{aligned} \tag{7.3.1}$$

for all $d_h^j \in V_{\mathrm{cdG}}^j$. The penalty parameter m is defined by [20]

$$m : \mathcal{E}_h \to \mathbb{R}, \quad \boldsymbol{x} \mapsto C_{\mathrm{pen}} \frac{\max\{n_{\mathcal{E}_h}^\top \mathbb{D}(u_h^{j,+}, \boldsymbol{x}) n_{\mathcal{E}_h}, n_{\mathcal{E}_h}^\top \mathbb{D}(u_h^{j,-}, \boldsymbol{x}) n_{\mathcal{E}_h}\}}{h}$$

and C_{pen} is chosen as in (4.4.7). The bilinear form for convection, production and injection is given by the non-standard form

$$\begin{aligned}&\mathcal{B}_{cq}(c_h^j, d_h^j; u_h^j) \\ &= \frac{1}{2}\sum_{E\in\mathcal{T}_h}\int_E (u_h^j\cdot\nabla_h c_h^j)d_h^j - c_h^j u_h^j\cdot\nabla_h d_h^j + (q^I + q^P)c_h^j d_h^j \,\mathrm{d}\boldsymbol{x} \\ &\quad + \frac{1}{2}\sum_{e\in\mathcal{E}_h^o}\int_e (u_h^j\cdot[\![c_h^j]\!])d_h^{j,*}\,\mathrm{d}s\end{aligned} \tag{7.3.2}$$

where $d_h^{j,*}$ is defined by

$$d_h^{j,*} = \begin{cases} d_h^{j,-} & \text{if } u_h^j\cdot n^+ > 0, \\ d_h^{j,+} & \text{if } u_h^j\cdot n^+ \le 0. \end{cases} \tag{7.3.3}$$

This formulation ensures that $\mathcal{B}_{cq}$ is semi-definite regardless of the properties of u_h^j. We do not need to restrict sums over edges to cells in $\mathcal{T}_{\mathrm{cG}}$ as in this region elements of V_{cdG} are continuous and therefore the jump terms on that region are 0. Also note that the dG method is a special case of the cdG method where $\mathcal{T}_{\mathrm{cG}} = \emptyset$. The formulation $\mathcal{B}_{cq}^{\mathrm{alt}}$ in Chapter 4 does not have the advantage of being semi-definite as in (7.3.2) but matches the original equation (1.1.3) more closely. We can move between the formulations using integration and the properties of $\nabla\cdot u$ as demonstrated in [20, Section 4].

As previously define $\mathcal{B}(c_h^j, d_h^j; u_h^j) := \mathcal{B}_d(c_h^j, d_h^j; u_h^j) + \mathcal{B}_{cq}(c_h^j, d_h^j; u_h^j)$ and

$$\mathcal{S}(c_h^j, d_h^j) := \sum_{e \in \mathcal{E}_{\mathrm{cG}}^j} \int_e M[\![c_h^j]\!] \cdot [\![d_h^j]\!] \, \mathrm{d}s \tag{7.3.4}$$

where

$$M := \left(C_d \frac{k^2}{h_e} \right).$$

Then for any $c_h^j, d_h^j \in V_{\mathrm{cdG}}^j$ and $x^j \in V_\perp^j$

$$\mathcal{S}(c_h^j, d_h^j) = \mathcal{S}(c_h^j, x^j) = \mathcal{S}(x^j, c_h^j) = 0.$$

We define the following norm for $u_h^j \in U$ and $c_h^j \in V_{\mathrm{dG}}^j$:

$$\begin{aligned} |||c_h^j|||^2 := & \sum_{E \in \mathcal{T}_h} \|\sqrt{\mathbb{D}(u_h^j)} \nabla_h c_h^j\|_{L^2(E)}^2 + \frac{1}{2} \|q_0 c_h^j\|_{L^2(\Omega)}^2 \\ & + \sum_{e \in \mathcal{E}_h^o} \frac{1}{2} \| |u_h^j \cdot n|^{1/2} [\![c_h^j]\!] \|_{L^2(e)}^2 + \sum_{e \in \mathcal{E}_h^o} \|\sqrt{m} [\![c_h^j]\!]\|_{L^2(e)}^2 \end{aligned} \tag{7.3.5}$$

where $q_0 := \sqrt{q^I + q^P}$. For $c_h^j \in V_{\mathrm{dG}}^j$ define

$$|c_h^j|_{\mathcal{S}}^2 := \sum_{e \in \mathcal{E}_{\mathrm{cG}}^j} \|\sqrt{M} [\![c_h^j]\!]\|_{L^2(e)}. \tag{7.3.6}$$

Notice that (7.3.6) is a semi-norm on V_{dG}^j but a norm on $V_\perp^j$.

Lemma 7.3.7. *If C_{pen} is chosen large enough then $\mathcal{B}$ is coercive for all $c_h^j \in V_{dG}^j$ and $u_h^j \in U$, i.e.,*

$$\mathcal{B}(c_h^j, c_h^j; u_h^j) \geq \Lambda_W |||c_h^j|||^2. \tag{7.3.8}$$

Proof. For $\mathcal{B}_d$ we have using Hölder's inequality

$$\begin{aligned} \mathcal{B}_d(c_h^j, c_h^j; u_h^j) = & \sum_{E \in \mathcal{T}_h} \|\sqrt{\mathbb{D}(u_h^j)} \nabla_h c_h^j\|_{L^2(E)}^2 + \sum_{e \in \mathcal{E}_h^o} \|\sqrt{m} [\![c_h^j]\!]\|_{L^2(e)}^2 \\ & - 2 \sum_{e \in \mathcal{E}_h^o} \int_e [\![c_h^j]\!] \cdot \{\!\{ \mathbb{D}(u_h^j) \nabla_h c_h^j \}\!\} \, \mathrm{d}s \end{aligned}$$

and then using the Cauchy-Schwarz inequality and an inverse inequality

$$\left|\int_e [\![c_h^j]\!] \cdot \{\!\{\mathbb{D}(u_h^j)\nabla_h c_h^j\}\!\}\right| = \frac{1}{2}\sum_{E\in\{E^+,E^-\}}\int_e \left(\sqrt{m}[\![c_h^j]\!]\right)\cdot\left(\frac{1}{\sqrt{m}}\{\!\{\mathbb{D}(u_h^j)\nabla_h c_h^j\}\!\}\right)\,\mathrm{d}s$$
$$\leq \frac{1}{2}\sum_{E\in\{E^+,E^-\}} \|\sqrt{m}[\![c_h^j]\!]\|_{L^2(e)}\cdot\|\sqrt{\mathbb{D}(u_h^j)}\nabla_h c_h^j\|_{L^2(E)}$$

for all $\mathcal{E}_h^o \ni e = E^+ \cap E^-$ provided C_{pen} is large enough. Using Young's inequality we combine each term in $|||c_h^j|||^2$.

For $\mathcal{B}_{cq}$ we have

$$\mathcal{B}_{cq}(c_h^j, c_h^j; u_h^j) = \frac{1}{2}\sum_{E\in\mathcal{T}_h}\int_E (q^I + q^P)(c_h^j)^2\,\mathrm{d}\boldsymbol{x} + \frac{1}{2}\sum_{e\in\mathcal{E}_h^o}\int_e |u_n^j\cdot n|[\![c_h^j]\!]\cdot[\![c_h^j]\!]\,\mathrm{d}s$$

where we have used $u_h^{j,+}\cdot n^+ = u_h^{j,-}\cdot n^+$ from the definition of U. Hölder's inequality completes the proof. □

We have by construction that $\mathcal{S}$ is coercive with constant one on $V_\perp^j$, i.e., for all $x_h^j \in V_\perp^j$

$$\mathcal{S}(x_h^j, x_h^j) = \|x_h^j\|_{\mathcal{S}}^2. \tag{7.3.9}$$

We discretize the time derivative with the backward Euler operator. Summing over each discrete time step gives

$$\begin{aligned}
&\sum_{j=1}^N\sum_{E\in\mathcal{T}_h}\int_E \varphi(\mathfrak{d}_{\mathrm{t}}c_h^j)c_h^j\,\mathrm{d}\boldsymbol{x}\\
&= \sum_{j=1}^N\sum_{E\in\mathcal{T}_h}\int_E \frac{\varphi}{\Delta t}\left(c_h^j c_h^j - c_h^{j-1}c_h^j\right)\,\mathrm{d}\boldsymbol{x}\\
&\geq \sum_{j=1}^N \frac{1}{\Delta t}\|\varphi^{1/2}c_h^j\|_{L^2(\Omega)}^2 - \frac{1}{2\Delta t}\left(\|\varphi^{1/2}c_h^{j-1}\|_{L^2(\Omega)}^2 + \|\varphi^{1/2}c_h^j\|_{L^2(\Omega)}^2\right)\\
&= \frac{1}{2\Delta t}\left(\|\varphi^{1/2}c_h^N\|_{L^2(\Omega)}^2 - \|\varphi^{1/2}c_h^0\|_{L^2(\Omega)}^2\right)
\end{aligned}$$

where we have used Young's inequality.

Definition 7.3.10. *Define the RT-dG approximation* $(u_h, p_h, c_\sigma) \in \Pi_{j=1}^N U \times \Pi_{j=1}^N P \times$

$\Pi_{j=1}^{N} V_{dG}^{j}$ *to* (1.1.3)-(1.1.8) *as that generated by the algorithm: For* $1 \leq j \leq N$ *and* $c_\sigma^{j-1} \in V_{dG}^{j}$ *find* $(u_h^j, p_h^j, c_\sigma^j) \in U \times P \times V_{dG}^{j}$ *such that*

$$(\nabla_h \cdot u_h^j, w_h^j) = (q^I - q^P, w_h^j), \tag{7.3.11}$$

$$(a^{-1}(c_\sigma^j)u_h^j, v_h^j) - (p_h^j, \nabla_h \cdot v_h^j) = (\rho(c_\sigma^j)g, v_h^j) \tag{7.3.12}$$

for all $(v_h^j, w_h^j) \in U \times P$ *and*

$$\sum_{E\in\mathcal{T}_h} \left(\int_E \varphi(\mathfrak{d}_t c_\sigma^j) d_h^j \, \mathrm{d}\boldsymbol{x} \right) + \mathcal{B}(c_\sigma^j, d_h^j; u_h^j) + \sigma \mathcal{S}(c_\sigma^j, d_h^j) = \sum_{E\in\mathcal{T}_h} \int_E \hat{c} q^I d_h^j \, \mathrm{d}\boldsymbol{x} \tag{7.3.13}$$

for all $d_h^j \in V_{dG}^{j}$.

Definition 7.3.14. *Define the RT-cdG approximation* $(u_h, p_h, c_h) \in \Pi_{j=1}^{N} U \times \Pi_{j=1}^{N} P \times \Pi_{j=1}^{N} V_{cdG}^{j}$ *to* (1.1.3)-(1.1.8) *as that generated by the algorithm: For* $1 \leq j \leq N$ *and* $c_h^{j-1} \in V_{cdG}^{j}$ *find* $(u_h^j, p_h^j, c_h^j) \in U \times P \times V_{cdG}^{j}$ *such that*

$$(\nabla_h \cdot u_h^j, w_h^j) = (q^I - q^P, w_h^j), \tag{7.3.15}$$

$$(a^{-1}(c_h^j)u_h^j, v_h^j) - (p_h^j, \nabla_h \cdot v_h^j) = (\rho(c_h^j)g, v_h^j) \tag{7.3.16}$$

for all $(v_h^j, w_h^j) \in U \times P$ *and*

$$\sum_{E\in\mathcal{T}_h} \left(\int_E \varphi(\mathfrak{d}_t c_h^j) d_h^j \, \mathrm{d}\boldsymbol{x} \right) + \mathcal{B}(c_h^j, d_h^j; u_h^j) + \sigma \mathcal{S}(c_h^j, d_h^j) = \sum_{E\in\mathcal{T}_h} \int_E \hat{c} q^I d_h^j \, \mathrm{d}\boldsymbol{x} \tag{7.3.17}$$

for all $d_h^j \in V_{cdG}^{j}$.

Theorem 7.3.18. *The solution* $c_\sigma \in \Pi_{j=1}^{N} V_{dG}^{j}$ *defined in Definition 7.3.10 converges to* $c_h \in \Pi_{j=1}^{N} V_{cdG}^{j}$ *defined in Definition 7.3.14 as* $\sigma \to \infty$, *i.e.,*

$$\lim_{\sigma\to\infty} (c_\sigma - c_h) = 0. \tag{7.3.19}$$

Proof. Following the argument of Section 7.1 we use Lemma 7.3.7 and (7.3.9). In order to complete the proof using this argument we must show that for every sequence

$\{c_i^j\}_i$ with elements in V_{dG}^j and $\lim_{i\to\infty} c_i^j = c^j \in V_{\mathrm{dG}}^j$ we have

$$\lim_{i\to\infty} \mathcal{B}(c_i^j, d_h^j; u^j(c_i^j)) = \mathcal{B}(c^j, d_h^j; u^j(c^j)) \qquad \forall d_h^j \in V_{\mathrm{dG}}^j \tag{7.3.20}$$

as in (7.1.11), where $u^j(\,\cdot\,)$ is the element in U solving (7.3.11)-(7.3.12) for a given element of V_{dG}^j. Note that $u^j : V_{\mathrm{dG}}^j \to U$ is a continuous map and so $\lim_{i\to\infty} u^j(c_i^j) = u^j(\lim_{i\to\infty} c_i^j) = u^j(c^j)$. This also holds for derivatives as they are taken piecewise. Therefore (7.3.20) holds at each timestep and for the whole discrete solution in time. □

7.4 Numerical Experiments with Super Penalization

We present numerical experiments to illustrate the results of this chapter. We focus on the behaviour of the approximations as $\sigma \to \infty$ on the whole domain, i.e., with $\mathcal{T}_h = \mathcal{T}_{\mathrm{cG}}$ for equations of non-negative characteristic form. For the equations of incompressible miscible displacement we introduce an algorithm to refine the $\mathcal{T}_h$ decomposition in time.

Equations of Non-Negative Characteristic Form

Example 7.4.1 *Let $\Omega = (0,1)^2$. We seek to solve*

$$-\varepsilon\Delta u + (1,1)\cdot\nabla u = f.$$

Given homogeneous Dirichlet boundary conditions f is chosen such that the solution is given by

$$u(x,y) := \left(x - \frac{\mathrm{e}^{(x-1)/\varepsilon} - \mathrm{e}^{-1/\varepsilon}}{1-\mathrm{e}^{-1/\varepsilon}}\right)\left(y - \frac{\mathrm{e}^{(y-1)/\varepsilon} - \mathrm{e}^{-1/\varepsilon}}{1-\mathrm{e}^{-1/\varepsilon}}\right).$$

For $0 < \varepsilon \ll 1$ this problem exhibits exponential boundary layers along the outflow boundaries $x = 1$ and $y = 1$ of width $\mathcal{O}(\varepsilon)$. We consider a uniformly refined

mesh of squares of edge length 2^{-4} and set $k = 1$ (piecewise bilinear polynomials).

We first look at an example without a layer by setting $\varepsilon = 10$. We set $\mathcal{T}_{\text{cG}} = \mathcal{T}_h$, i.e., the cG method. Figure 7.4.1 shows the behaviour of the difference between the dG and cG approximations in the $L^2(\mathcal{T}_h)$ norm, $H^1(\mathcal{T}_h)$ semi-norm and the L^2 norm of the jumps across edges (represented by $[\![\cdot]\!]$). As σ grows the difference in each norm decreases linearly. The jumps in either approximation are already very small, i.e., the dG approximation is very close to an element in the cG space. We do not see oscillations polluting the continuous approximation.

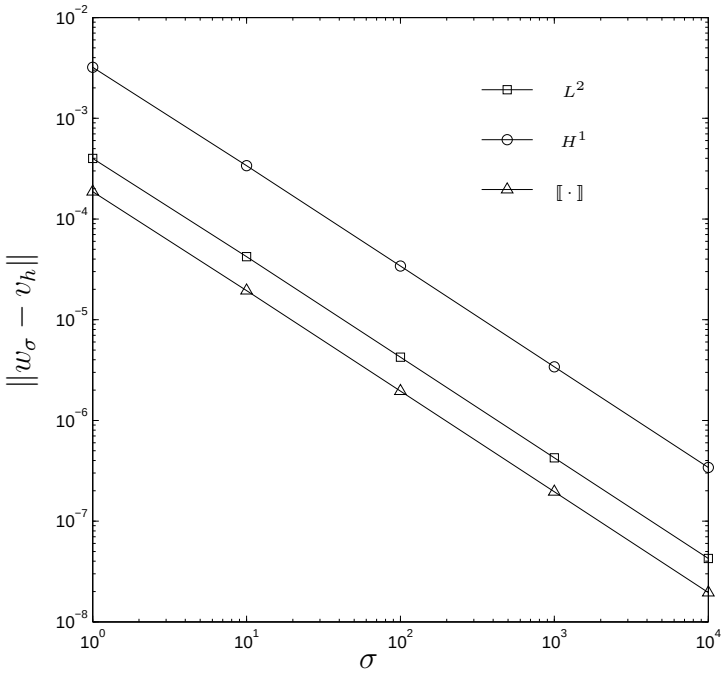
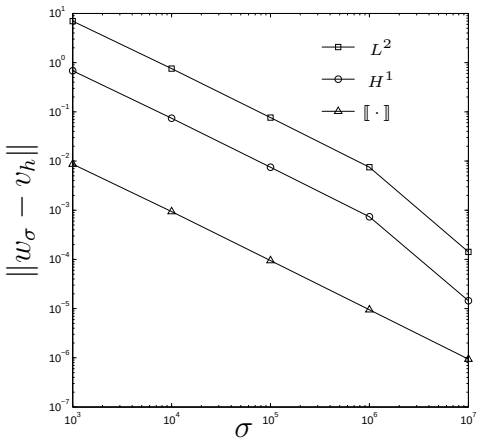
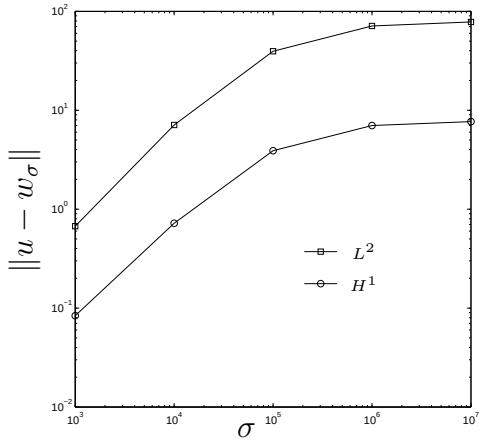


Figure 7.4.1: Example 7.4.1 with $\varepsilon = 10$ and $\mathcal{T}_{\text{cG}} = \mathcal{T}_h$. As the penalty parameter is increased the difference between the cG and dG approximations decreases linearly in the given norms.

We now motivate the cdG method by choosing $\varepsilon = 10^{-4}$ and again setting $\mathcal{T}_{\text{cG}} = \mathcal{T}_h$. The example now has a sharp layer at the outflow boundaries. We see in Figure 7.4.2(a) that increasing σ gives a linear response to the error as in Figure 7.4.1. When we look at the error in the dG approximation in Figure 7.4.2(b) we see that the approximation becomes worse as the penalty is increased. The layer causes non-physical oscillations to pollute the approximation. Although we see convergence of the dG approximation to the cG approximation this property is not desirable.



(a) The difference between the cG and dG approximations.



(b) The error in the dG approximation.

Figure 7.4.2: Example 7.4.1 with $\varepsilon = 10^{-4}$ and $\mathcal{T}_{\text{cG}} = \mathcal{T}_h$. Now the problem has a layer the error in the dG approximation grows as σ is increased. Non-physical oscillations pollute the approximation.

Example 7.4.2 *Let $\Omega = (0,1)^2$. We seek to solve*

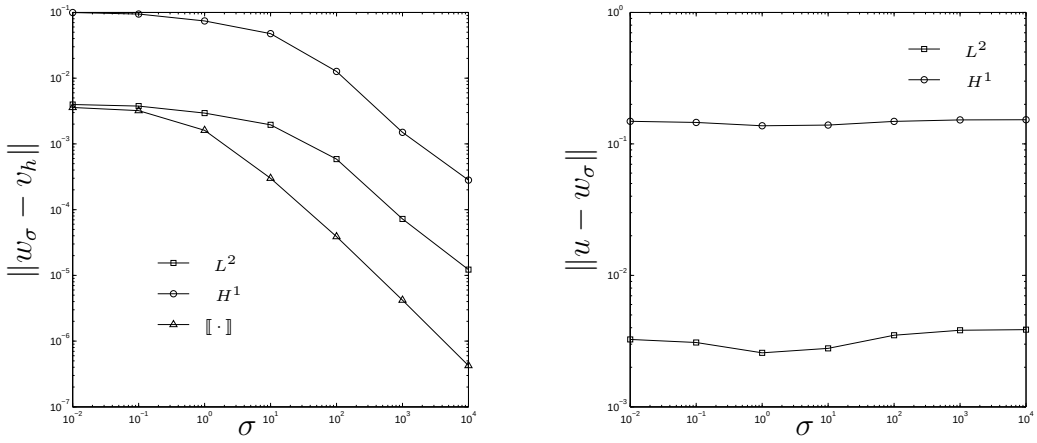
$$(2 - y^2, 2 - x) \cdot \nabla u + \left(1 + (1 + x)(1 + y)^2\right) = f.$$

The inflow Dirichlet boundary conditions and f are chosen such that the solution is given by

$$u(x, y) := 1 + \sin\left(\frac{\pi}{8}(1 + x)(1 + y)^2\right).$$

This example is taken from [30, 82]. The solution does not exhibit layers. We consider a uniformly refined mesh of squares of edge length 2^{-4} and set $\mathcal{T}_h = \mathcal{T}_{\text{cG}}$ and $k = 1$. Following Remark 7.2.7 we set $C_d = 0$ and $C_{ar} = 1$. We plot the difference between the cG and dG approximations as σ is increased in Figure 7.4.3(a). For small values of σ the difference between approximations is not overly affected by increasing the penalty parameter. This is due to the penalization of the jump terms coming from $\||b \cdot n|^{1/2}[\![w]\!]\|_{L^2(\mathcal{E})}$ in (7.2.9), the same term which enabled the analysis in Chapter 3. The additional penalization due to σ does not significantly increase the size of the penalty. As σ becomes larger the contribution to the penalization becomes significant relative to the advection term and we see again the linear decrease with increasing σ.

In Figure 7.4.3(b) we plot the error in the dG approximation with increasing σ. We do not see the same behaviour as in Figure 7.4.2(b) as the cG approximation does not suffer from the same non-physical oscillations as it does for singularly perturbed problems. There is however a slight dip in the error around $\sigma = 1$ corresponding to the optimum amount of penalization, i.e., the dG approximation benefits from some restriction on the size of the jumps.



(a) The difference between the cG and dG approximations.

(b) The error in the dG approximation.

Figure 7.4.3: Example 7.4.2 with $\mathcal{T}_{\text{cG}} = \mathcal{T}_h$. The increase in error associated with the non physical oscillations of the cG approximation is not present as the problem exhibits no layers.

We finally note that these results do not suggest that the cG method should be chosen over the dG method for hyperbolic problems like Example 7.4.2. Other factors must also be considered. For example with refinement of the mesh it can be shown [99] that the standard cG method has an order of convergence of $\mathcal{O}(h^k)$ in L^2 compared to $\mathcal{O}(h^{k+1/2})$ for the dG method, e.g., [37].

Equations of Incompressible Miscible Displacement

As well as verifying Theorem 7.3.18 we wish to show that if the region where continuous elements are used is chosen appropriately there is little difference in the approximations via the RT-cdG or RT-dG method (where the concentration is approximated in the dG space).

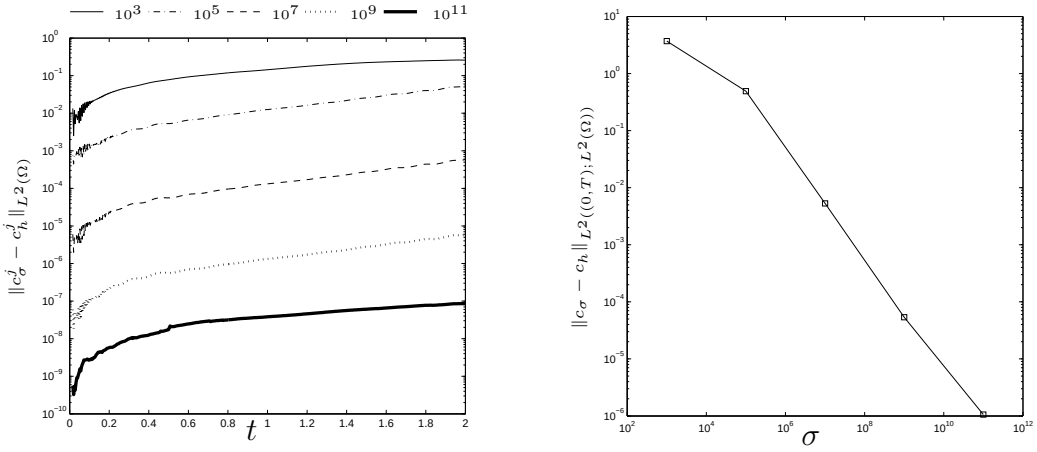
We study a standard example [20, 50, 123] to illustrate the performance of the cdG method for the incompressible miscible displacement problem (1.1.3)-(1.1.8).

Example 7.4.3 *Let $\Omega = (0,1)^2$. The injection (resp. extraction) well is located at $(1,1)$ (resp. $(0,0)$). We represent the injection (resp. extraction) term by a function which is constant on the element including the injection (resp. extraction) point, and zero elsewhere, such that $\int_\Omega q^I \,\mathrm{d}\boldsymbol{x} = \int_\Omega q^P \,\mathrm{d}\boldsymbol{x} = 0.018$. In (4.2.1) we set $d_l = 1.8 \times 10^{-4}$, $d_m = 1.8 \times 10^{-6}$ and $d_t = 1.8 \times 10^{-5}$. The porosity is set to 0.1. The concentration dependent viscosity is given by $\mu(c) = \mu(0)(1 + (\mathcal{M}^{1/4} - 1)c)^{-4}$ where $\mathcal{M} = 41.0$ is the mobility ratio (the ratio of the viscosity of the fluids), and $\mu(0) = 1$. This commonly used relation is called the quarter-power mixing rule, e.g., [123, 124]. For the initial concentration we set $c_0 = 0$ corresponding to Ω uniformly filled with one fluid. Set $\mathbb{K} = 0.0288\mathbb{I}$.*

We consider a uniform refinement of Ω into squares of side $h = 2^{-4}$ with timestep 4×10^{-3} and time interval $(0.0, 2.0)$. We use the lowest order RT elements, piecewise constant approximation space for pressure and bilinear polynomials to approximate concentration. With these values a sharp front in the concentration component spreads from the injection to extraction point. As can be seen in Figure 7.4.6(d) this causes oscillations in the continuous approximation.

First we present the difference between the dG approximation and the cG approximation (i.e., with $\mathcal{T}_{\mathrm{cG}} = \mathcal{T}_h$) as $\sigma \to \infty$. In Figure 7.4.4 we show $\|c_\sigma - c_h\|$ in both the L^2 norm against time and the $L^2((0,T); L^2(\Omega))$ norm against increasing σ. In Figure 7.4.4(a) we see a sharp increase in the error over the first few iterations. The initial conditions are in the continuous approximation space so the cG and dG approximations are close. As the layer spreads through the domain the difference between the cG and dG approximations for a given σ in the L^2 norm increases slowly. This is because the number of edges in the vicinity of the layer increases. Figure 7.4.4(b) shows the same behaviour as the stationary Examples 7.4.1 and 7.4.2.

Picking $\mathcal{T}_{\mathrm{cG}}$ in the examples in Chapter 3 was done via knowledge of the true solution and hence knowledge of any layers. We do not have this luxury for the problem considered in this section. We therefore undertake the following procedure for determining $\mathcal{T}_{\mathrm{cG}}$:



(a) Evolution of the difference between the cG and dG approximations for $\sigma = 10^3$ to 10^{11} in L^2 norm.

(b) Plot of the difference between the cG and dG approximations in the $L^2(L^2)$ norm as σ is increased.

Figure 7.4.4: The effect of increasing σ for Example 7.4.3 with $\mathcal{T}_{\text{cG}} = \mathcal{T}_h$.

(1) Determine the initial pressure and velocity given c_h^0 and the injection profile.

(2) Solve for the first time step using a RT-dG method to find a discontinuous c_h^1 (and also p_h^1 and u_h^1).

(3) For all edges determine $\|[\![c_h]\!]\|_{L^2(e)}$.

(4) Flag every cell where each edge satisfies $\|[\![c_h]\!]\|_{L^2(e)} < \texttt{tol}$.

(5) If every edge of a cell is flagged set that cell to be part of $\mathcal{T}_{\text{cG}}$ in the next iteration. Otherwise the element will be in $\mathcal{T}_{\text{dG}}$.

(6) For n iterations use the cdG mesh defined in the previous step.

(7) For the $(n+1)^{th}$ iteration reset the mesh to be entirely dG, i.e., $\mathcal{T}_{\text{cG}}^{n+1} = \emptyset$ for the concentration component, then return to step (3).

The number of iterations between each cdG refinement and the tolerance should consider the expected motion of the fluid and the time step. We do not consider increasing σ for the cdG method, but rather study the performance of the method as the tolerance is increased by comparing the cdG approximation with a dG approximation where $C_{\text{pen}} = 10$ and $\sigma = 0$. With these parameters we set the number of iterations between redefining the cdG space to be 5.

In Figure 7.4.5 we see that as the tolerance is decreased the difference between the dG and cdG approximations in the L^2 norm gets smaller. With a smaller tolerance fewer cells are marked as being continuous. The difference introduced by using some continuous elements does not seem to propagate in time.

In Table 7.4.1 we see that the number of degrees of freedom saved over the simulation (500 steps with $T = 2.0$, $\Delta t = 4 \times 10^{-3}$) is considerable. The effect on the approximation is however small measured in the $L^2(L^2)$ norm. The number of degrees of freedom for the cG method is not 128,000 as would be expected (one degree of freedom per vertex on a 16×16 square mesh for 500 timesteps) due to every fifth iteration being discontinuous.

In Figure 7.4.6 we show the dG, cG and cdG approximations after 380 timesteps. There is no visible difference between the plots for dG and cdG at each tolerance (Figures 7.4.6 (a), (b) and (c)). However for the fully continuous approximation the oscillations induced by the layer are clearly visible and distort the plot.

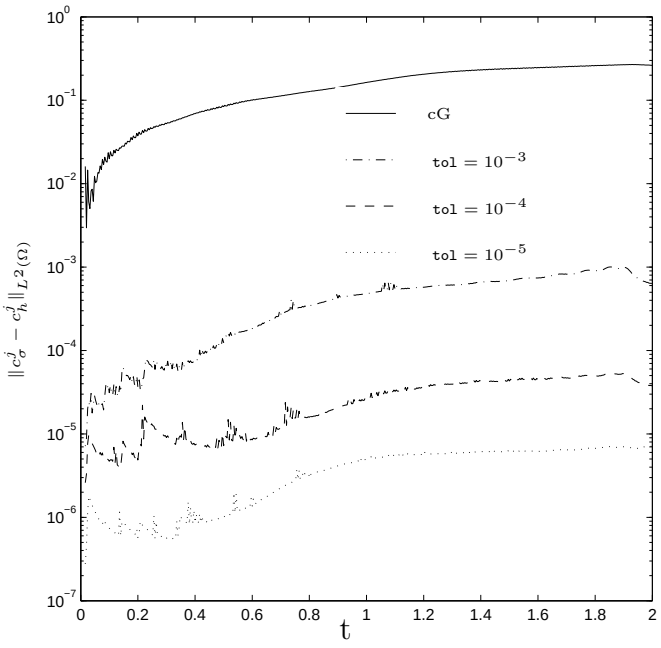
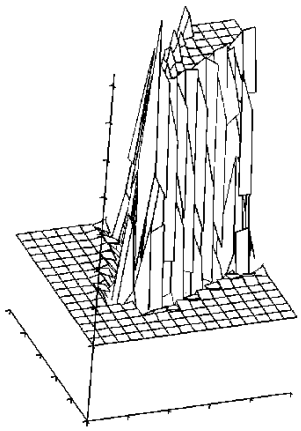


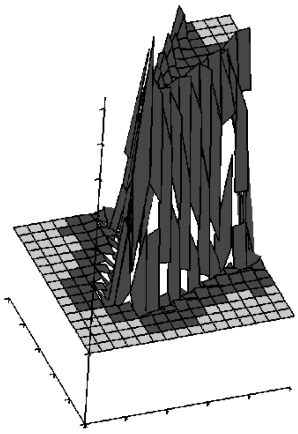
Figure 7.4.5: The behaviour of the cdG approximation for Example 7.4.3. Using some continuous elements does not dramatically increase the error of the cdG approximation compared to the dG approximation.

tol	dofs	$\|c_\sigma - c_h\|_{L^2((0,T);L^2(\Omega))}$
cG	219,470	3.9970×10^{0}
10^{-3}	323,488	1.2073×10^{-2}
10^{-4}	355,328	7.0904×10^{-4}
10^{-5}	382,384	1.0455×10^{-4}
dG	512,000	0.0000×10^{0}

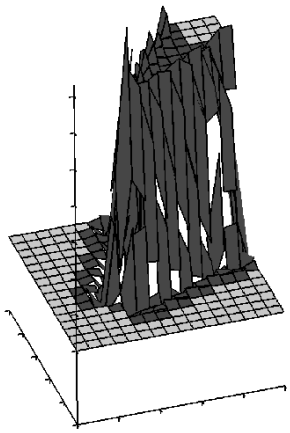
Table 7.4.1: The number of degrees of freedom used for 500 timesteps in Example 7.4.3. When $\texttt{tol} = 10^{-5}$ only 74.6% of the degrees of freedom are used compared to 43% for the continuous approximation.



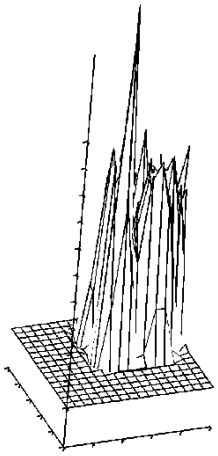
(a) The fully discontinuous approximation.



(b) The cdG approximation with `tol` $= 10^{-4}$.



(c) The cdG approximation with `tol` $= 10^{-3}$.



(d) The fully continuous approximation.

Figure 7.4.6: A plot of the cG method, cdG method and dG method at time 1.52 (380 time steps) for Example 7.4.3. The discontinuous region is marked in dark grey for the cdG method. There is no appreciable difference between the first three plots. The oscillations are clearly visible in the fully continuous plot.

Chapter 8

On Determining the $\mathcal{T}_h$ Decomposition using the Discontinuous Galerkin Approximation

The application of the analysis in Chapter 3 is limited by the assumptions made in that chapter and the ability to identify the Ω decomposition. The work in Chapter 7 provides insight into the role of the degrees of freedom in the cG and dG methods. However the super penalty method does not reduce the number of degrees of freedom used. We therefore seek a practical approach to determining the $\mathcal{T}_h$ decomposition.

Recall that the dG approximation to the advection diffusion reaction problem has better stability properties than the cG approximation (although streamline terms can be added to the dG method to show an inf-sup stability result as in, e.g., [41]). It is of note that in general dG approximations have small interelement jumps away from the presence of layers in the solution. In this chapter we show that where the interior penalty dG approximation to a given ADR problem has small jumps we can replace the discontinuous elements with continuous elements at a cost related to the size of the jumps removed. We can use this approach to determine the best $\mathcal{T}_h$ decomposition for the problem (in the sense of reduction in degrees of freedom). This offers a practical improvement over the approach of Chapter 3 although we

have not been able to show a stability result using this approach.

8.1 Continuity and Coercivity

The continuity and coercivity coefficients (see Section 2.1 for a definition) may depend on the parameters of the problem, in particular on ε and h. If this were not the case we could formulate an ε independent stability result using the Lax-Milgram theorem and an error bound using Céa's lemma, which we know is not possible for the continuous bilinear form (cf., (2.1.11)). To circumvent this problem for the dG method it is possible to show an inf-sup stability result [10, 41].

We present a detailed analysis of the continuity and coercivity constants for the interior penalty dG bilinear form. If we know the behaviour of the continuity and coercivity coefficients with respect to ε, h and the other parameters we can devise a computational method for selecting $\mathcal{T}_h$.

We assume the mesh $\mathcal{T}_h$ is shape regular with regularity constant C_{reg} as in (1.3.8). We also recall the following result found by combining trace and inverse inequalities (cf., Corollary 1.3.13):

$$\|\nabla_h w\|^2_{L^2(e)} \leq C_{\text{ti}} h_E^{-1} \|\nabla_h w\|^2_{L^2(E)}.$$

Lemmas 8.1.1, 8.1.4 and 8.1.8 are taken from [75]. We include the proofs here to fully present the origin of the coefficients. Throughout this chapter $|||\cdot|||$ refers to the norm defined in (2.2.9).

Lemma 8.1.1. *There exists $\overline{\sigma}$ such that for every $\sigma \geq \overline{\sigma}$ we have*

$$\mathcal{B}_\varepsilon(w,w) \geq \Lambda_{\text{cc}} |||w|||^2 \qquad \forall w \in V_{dG}. \tag{8.1.2}$$

Specifically $\overline{\sigma} = 0$ for $\vartheta = 1$ with $\Lambda_{\text{cc}} = 1$ and $\overline{\sigma} = 4C_{\text{ti}}/C_{\text{reg}}$ for $\vartheta = -1$ with $\Lambda_{\text{cc}} = 1/2$.

Proof. Using the definitions of the L^2 norm and $\mathcal{B}_\varepsilon$ we have

$$\begin{aligned}\mathcal{B}_\varepsilon(w,w) = &\sum_{E\in\mathcal{T}_h} \varepsilon\|\nabla_h w\|^2_{L^2(E)} + \sum_{e\in\mathcal{E}_h} \frac{\sigma\varepsilon}{h_e}\|[\![w]\!]\|^2_{L^2(e)} + \|r^{1/2}w\|^2_{L^2(\Omega)} \\ &+ \sum_{e\in\mathcal{E}_h} \frac{1}{2}\||b\cdot n|^{1/2}[\![w]\!]\|^2_{L^2(e)} + (\vartheta-1)\sum_{e\in\mathcal{E}_h}\int_e \varepsilon\{\!\{\nabla_h w\}\!\}\cdot[\![w]\!]\,\mathrm{d}s\end{aligned}$$

where the advection reaction terms follow from integration by parts as in (3.2.21).

Thus when $\vartheta = 1$ coercivity holds with $\Lambda_{\mathrm{cc}} = 1$. For E^+, E^- as two cells sharing and edge e, when $\vartheta = -1$ we use the definition of the average to show

$$\begin{aligned}&2\int_e \varepsilon\{\!\{\nabla_h w\}\!\}\cdot[\![w]\!]\,\mathrm{d}s \\ &= \sum_{E\in\{E^+,E^-\}}\int_e \varepsilon\nabla_h w\cdot[\![w]\!]\,\mathrm{d}s \\ &\le \sum_{E\in\{E^+,E^-\}} C_{\mathrm{ti}}^{1/2}\varepsilon h_E^{-1/2}\|\nabla_h w\|_{L^2(E)}\|[\![w]\!]\|_{L^2(e)} \\ &\le \frac{1}{2}\sum_{E\in\{E^+,E^-\}}\left(\varepsilon\|\nabla_h w\|^2_{L^2(E)}\right)^{1/2}\left(\frac{\sigma\varepsilon}{h_e}\|[\![w]\!]\|^2_{L^2(e)}\right)^{1/2} \\ &\le \frac{\varepsilon}{8}\|\nabla_h w\|^2_{L^2(E^+)} + \frac{\varepsilon}{8}\|\nabla_h w\|^2_{L^2(E^-)} + \frac{1}{2}\frac{\sigma\varepsilon}{h_e}\|[\![w]\!]\|^2_{L^2(e)},\end{aligned} \tag{8.1.3}$$

where in the third step we have used the condition on the size of σ. We combine each of these terms with $|||w|||^2$, the final term being the determining factor. Therefore coercivity holds with $\Lambda_{\mathrm{cc}} = 1/2$ when $\vartheta = -1$. □

We would like to be able to show continuity *independently of* ε. For the advection diffusion reaction problem this is not generally the case. However in our setting we do not allow $h \to 0$ unless $\varepsilon \to 0$ (as the Péclet number must be greater than 1 via Assumption 2.3.9) and therefore we formulate the continuity independently of ε (but depending on h) provided $\rho > 0$.

Lemma 8.1.4. *Provided Assumption 2.1.5 holds and we have the conditions on the size of σ from Lemma 8.1.1 then*

$$|\mathcal{B}_\varepsilon(w,\hat{w})| \le \Lambda_{\mathrm{ct}}|||w|||||\hat{w}||| \qquad \forall w,\hat{w}\in V_{dG} \tag{8.1.5}$$

where

$$\Lambda_{\mathrm{ct}} := C \max_{E\in\mathcal{T}_h} \left\{ 1, \left\| \frac{b}{\rho h_E} \right\|_{L^\infty(E)} \right\} \tag{8.1.6}$$

with C independent of the coefficients of $\mathcal{B}_\varepsilon$ and mesh parameters but depending on the shape regularity via the trace and inverse inequalities.

Proof. We rewrite the bilinear form by integrating the advection term on the cell by parts and making use of the identity (1.3.25)

$$\begin{aligned}
\mathcal{B}_\varepsilon(w,\hat{w}) = &\sum_{E\in\mathcal{T}_h} \int_E \varepsilon \nabla_h w \cdot \nabla_h \hat{w} + (c - \frac{1}{2}\nabla_h \cdot b) w\hat{w} \,\mathrm{d}\boldsymbol{x} \\
&+ \frac{1}{2} \sum_{E\in\mathcal{T}_h} \int_E (b\cdot\nabla_h w)\hat{w} - (b\cdot\nabla_h \hat{w}) w \,\mathrm{d}\boldsymbol{x} \\
&+ \sum_{e\in\mathcal{E}_h} \int_e \left(\frac{1}{2}|b\cdot n| + \frac{\sigma\varepsilon}{h_e} \right) [\![w]\!] \cdot [\![\hat{w}]\!] \,\mathrm{d}s \\
&+ \sum_{e\in\mathcal{E}_h} \int_e \vartheta\varepsilon \{\!\{ \nabla_h \hat{w} \}\!\} \cdot [\![w]\!] - \varepsilon \{\!\{ \nabla_h w \}\!\} \cdot [\![\hat{w}]\!] \,\mathrm{d}s \\
&+ \sum_{e\in\mathcal{E}_h^o} \int_e [\![\hat{w}]\!] \cdot \{\!\{ bw \}\!\} - [\![w]\!] \cdot \{\!\{ b\hat{w} \}\!\} \,\mathrm{d}s \\
= &\,\mathrm{I} + \mathrm{II} + \mathrm{III} + \mathrm{IV} + \mathrm{V}.
\end{aligned} \tag{8.1.7}$$

For term IV we have, following the steps of (8.1.3) from Theorem 8.1.1, that

$$\int_e \varepsilon \{\!\{ \nabla_h w \}\!\} \cdot [\![\hat{w}]\!] \,\mathrm{d}s \le \frac{1}{4} \sum_{E\in\{E^+,E^-\}} \left(\varepsilon \|\nabla_h w\|^2_{L^2(E)} \right)^{1/2} \left(\frac{\sigma\varepsilon}{h_e} \|[\![\hat{w}]\!]\|^2_{L^2(e)} \right)^{1/2}$$

and similarly with w, $\hat{w}$ interchanged. For term II we have, using an inverse inequality,

$$\begin{aligned}
\left| \int_E (b\cdot\nabla_h w)\hat{w} \,\mathrm{d}\boldsymbol{x} \right| &\le \|b\|_{L^\infty(E)} \|\nabla_h w\|_{L^2(E)} \|\hat{w}\|_{L^2(E)} \\
&\le C \left\| \frac{b}{\rho h_E} \right\|_{L^\infty(E)} \|r^{1/2} w\|_{L^2(E)} \|r^{1/2}\hat{w}\|_{L^2(E)}
\end{aligned}$$

and similarly with the terms interchanged. For each part of V we have

$$\left|\int_e [\![w]\!]\cdot\{\!\{b\hat{w}\}\!\}\,\mathrm{d}s\right| \leq \|b\|_{L^\infty(e)}\|[\![w]\!]\|\|\{\!\{\hat{w}\}\!\}\|_{L^2(e)}$$
$$\leq C\sum_{E\in\{E^+,E^-\}}\left\|\frac{b}{\rho h_E}\right\|_{L^\infty(E)}\|r^{1/2}w\|_{L^2(E)}\|r^{1/2}\hat{w}\|_{L^2(E)}.$$

For terms I and III we simply apply the Cauchy-Schwarz inequality. The proof is concluded by summing up each of the terms and bounding by $|||\cdot|||$. □

If $r\equiv 0$ then we may not proceed in this manner and the continuity is formulated with an adverse dependence on ε.

Lemma 8.1.8. *If $r(\boldsymbol{x})\equiv 0$ on Ω and we have the conditions on the size of σ from Theorem 8.1.1 then*

$$|\mathcal{B}_\varepsilon(w,\hat{w})|\leq\Lambda_{\mathrm{ct}}|||w|||||\hat{w}||| \qquad \forall w,\hat{w}\in V_{dG} \tag{8.1.9}$$

where

$$\Lambda_{\mathrm{ct}}:=\frac{C}{\varepsilon}\max_{E\in\mathcal{T}_h}\left\{1,\|b\|_{L^\infty(E)}\right\} \tag{8.1.10}$$

where C is independent of ε and h but depends on the mesh regularity.

Proof. Using the formulation (8.1.7) we may bound terms I, III and IV in the same way as in Theorem 8.1.4 noting that the second part of I is zero. For term V we have

$$\left|\int_e [\![\hat{w}]\!]\cdot\{\!\{bw\}\!\}\right| \leq \int_e\left|\frac{\varepsilon^{1/2}}{h_e^{1/2}}[\![w]\!]\right|\left|\frac{b}{\varepsilon^{1/2}}\right|\left|h_e^{1/2}\{\!\{\hat{w}\}\!\}\right|\,\mathrm{d}s$$
$$\leq\sum_{E\in\{E^+,E^-\}}\frac{1}{\varepsilon^{1/2}}\|b\|_{L^\infty(E)}\left(\frac{\sigma\varepsilon}{h_e}\|[\![w]\!]\|^2_{L^2(e)}\right)^{1/2}\left(\|\hat{w}\|^2_{L^2(E)}\right)^{1/2}$$

with a similar result for the other part of the term. For II we have

$$\left|\int_E(b\cdot\nabla_h w)\hat{w}\,\mathrm{d}\boldsymbol{x}\right|\leq\frac{1}{\varepsilon^{1/2}}\|b\|_{L^\infty(E)}\left(\varepsilon\|\nabla_h w\|^2_{L^2(E)}\right)^{1/2}\left(\|\hat{w}\|^2_{L^2(E)}\right)^{1/2}.$$

Summing over each element of the mesh we then use the Poincaré Friedrichs inequality as in [33, (1.8)] to show

$$\|w\|_{L^2(\Omega)} \leq \frac{C}{\varepsilon^{1/2}} \left(\sum_{E \in \mathcal{T}_h} \varepsilon \|\nabla_h w\|^2_{L^2(E)} + \sum_{e \in \mathcal{E}_h} \frac{\varepsilon \sigma}{h_e} \|[\![w]\!]\|^2_{L^2(e)} \right)^{1/2}. \tag{8.1.11}$$

Summing up each of the terms and using the Cauchy-Schwarz inequality gives

$$\mathcal{B}_\varepsilon(w, \hat{w}) \leq \frac{C}{\varepsilon} \max_{E \in \mathcal{T}_h} \{1, \|b\|_{L^\infty(E)}\} \left(|||w|||^2 + \|w\|^2_{L^2(\Omega)} \right)^{1/2} \left(|||\hat{w}|||^2 + \|\hat{w}\|^2_{L^2(\Omega)} \right)^{1/2}$$

and combining this with (8.1.11) concludes the proof. □

Remark 8.1.12. *Lemma 8.1.8 may give a smaller coercivity constant that Lemma 8.1.4. For our problems of interest, however, we consider $\varepsilon \ll h$ and sharpening layers as $\varepsilon \to 0$ for a fixed h, and $\rho > 0$, so we make the assumption that the bound of Lemma 8.1.4 is smaller.*

8.2 Determining the $\mathcal{T}_h$ Decomposition

Define by w_h and v_h the discontinuous and continuous discontinuous Galerkin approximations to the advection diffusion reaction equation as defined by (2.2.8) and (2.3.6) (for a given $\mathcal{T}_{\text{cG}}$). By subtraction we have the following orthogonality result:

$$\mathcal{B}_\varepsilon(w_h - v_h, v) = 0 \qquad \forall v \in V_{\text{cdG}}. \tag{8.2.1}$$

Now suppose we decompose the cdG approximation by $v_h = v_m + v_k$ where v_m, $v_k \in V_{\text{cdG}}$. With the orthogonality result (8.2.1) and the coercivity and continuity of the previous section we may show

$$\begin{aligned} \Lambda_{\text{cc}} |||w_h - v_h|||^2 &\leq \mathcal{B}_\varepsilon(w_h - v_h, w_h - v_h) \\ &= \mathcal{B}_\varepsilon(w_h - v_h, w_h - v_m) - \mathcal{B}_\varepsilon(w_h - v_h, v_k) \\ &\leq \Lambda_{\text{ct}} |||w_h - v_h||| |||w_h - v_m|||. \end{aligned}$$

The decomposition of v_h is not unique. For a fixed $\mathcal{T}_{\text{cG}}$ we may rearrange the above

result to give

$$|||w_h - v_h||| \leq \frac{\Lambda_{\text{ct}}}{\Lambda_{\text{cc}}} \min_{v_m \in V_{\text{cdG}}} |||w_h - v_m|||. \tag{8.2.2}$$

This is of course just a modified form of Céa's lemma.

We seek to answer, therefore, two questions. Firstly for a given $\mathcal{T}_{\text{cG}}$ how small can we expect $|||w_h - v_m|||$ to be? With an answer to this question we can then proceed to ask if it is possible (or practicable) to choose $\mathcal{T}_{\text{cG}}$ when w_h is known such that we have both $|||w_h - v_m|||$ small and $\mathcal{T}_{\text{cG}}$ relatively large compared to $\mathcal{T}_h$.

We introduce the Oswald interpolation operator [109].

Definition 8.2.3. *Denote by $\mathcal{V}_h$ the set of vertices of a mesh $\mathcal{T}_h$. For any $w \in V_{dG}$ the Oswald interpolation operator $\mathcal{O}s(w) : V_{dG} \to V_{cG}$ is defined as follows: For all $\nu \in \mathcal{V}_h$*

$$\mathcal{O}s(w(\nu)) = \frac{1}{n_\nu} \sum_{E \in \Delta_\nu} w|_E(\nu) \tag{8.2.4}$$

if $\nu \notin \partial\Omega$ and equal to the boundary conditions if $\nu \in \partial\Omega$. Here Δ_ν denotes the set of cells containing ν (the patch of ν) and n_ν the cardinality of the set Δ_ν, i.e., the number of cells in the patch.

The Oswald interpolant is therefore an averaging type operator. We have the following theorem [89, 90] concerning the error of the interpolation.

Theorem 8.2.5. *Let $\mathcal{T}_h$ be a conforming mesh with boundary Γ. Let g be the restriction to Γ of some function in V_{cG}. Then for any $w \in V_{dG}$ and multi index α with $|\alpha| = 0, 1$ the following approximation result holds: There exists $\mathcal{O}s(w) \in V_{cG}$ satisfying $\mathcal{O}s(w)|_\Gamma = g$ and*

$$\begin{aligned} &\sum_{E \in \mathcal{T}_h} \|D^\alpha(w - \mathcal{O}s(w))\|^2_{L^2(E)} \\ &\leq C_m \left(\sum_{e \in \mathcal{E}_h^o} h_e^{1-2|\alpha|} \|[\![w]\!]\|^2_{L^2(e)} + \sum_{e \in \Gamma} h_e^{1-2|\alpha|} \|w - g\|^2_{L^2(e)} \right) \end{aligned}$$

where C_m is independent of w and h.

Proof. See [89, Theorem 2.2] and the extensions in [90, Section 2]. □

Theorem 8.2.5 shows that when approximating discontinuous piecewise polynomials with continuous piecewise polynomials (of the same degree) the error is dependent on the jumps of the discontinuous function. It is also possible to formulate the theorem for non-conforming meshes. The following corollary applies Theorem 8.2.5 to cdG spaces.

Corollary 8.2.6. *Let $\mathcal{T}_{cG}$ be the continuous region of $\mathcal{T}_h$, with Γ_{cG} the boundary of $\mathcal{T}_{cG}$ and Γ the boundary of $\mathcal{T}_h$. Let g be the restriction to Γ_{cG} of some function in $V_{cG}(\mathcal{T}_{cG})$ which is zero on $\Gamma \cap \Gamma_{cG}$. For all $w \in V_{dG}(\mathcal{T}_{cG})$ and multi index α with $|\alpha| = 0, 1$ the following approximation result holds: There exists $\mathcal{O}s(w) \in V_{cdG}(\mathcal{T}_{cG})$ which is zero on $\Gamma \cap \Gamma_{cG}$ satisfying*

$$\begin{aligned} &\sum_{E\in\mathcal{T}_{cG}} \|D^\alpha(w - \mathcal{O}s(w)\|^2_{L^2(E)} \\ &\leq C_m \left(\sum_{e\in\mathcal{E}_{cG}} h_e^{1-2|\alpha|} \|[\![w]\!]\|^2_{L^2(e)} + \sum_{e\in J} h_e^{1-2|\alpha|} \|w^{\mathcal{C}} - g\|^2_{L^2(e)} \right) \end{aligned} \tag{8.2.7}$$

where C_m is independent of w and h and $w^{\mathcal{C}}$ is the trace of w on the continuous side of J.

Proof. Recall that by definition $\mathcal{E}_{\mathrm{cG}}$ does not include J but does include the exterior boundary. We therefore apply Theorem 8.2.5 and use the fact that $g = 0$ on the exterior boundary and the definition of the jump for $e \in \Gamma$. □

We now wish to specify g on J. To apply Corollary 8.2.6 we see that on any points where J meets the exterior boundary we must have $g = 0$. Where $\mathcal{V}_e$ is the set of vertices of $\mathcal{T}_{\mathrm{cG}}$ on the interior of J, denoted ν_e, (i.e., excluding vertices on the exterior boundary) we specify

$$g(w(\nu_e)) = \frac{1}{n_{\nu_e}} \sum_{e\in\Delta_{\nu_e}} w^{\mathcal{C}}|_e(\nu_e)$$

where Δ_{ν_e} denotes the set of edges of $\mathcal{T}_{\mathrm{cG}}$ containing ν_e and n_{ν_e} the cardinality of Δ_{ν_e}, and $w^{\mathcal{C}}$ denotes the trace of w from the continuous side of the edge. In this

manner g is the average of the values of $w^{\mathcal{C}}$ at each vertex on the interior of J. We can therefore modify Theorem 8.2.5 in one dimension to show for this g

$$\sum_{e\in J}\|w^{\mathcal{C}}-g\|^2_{L^2(e)} \leq C_m \sum_{\substack{\text{vertices}\\ \text{on J}}} \overline{h}_e[w^{\mathcal{C}}]^2, \tag{8.2.8}$$

where $[w^{\mathcal{C}}] = |w^{\mathcal{C},+} - w^{\mathcal{C},-}|$ on the interior of J (i.e., the one dimensional jump at each vertex), $[w^{\mathcal{C}}] = |w^{\mathcal{C}}|$ at the exterior boundary and $\overline{h}_e$ is the average of the diameter of the edges meeting at the vertex. We see that smaller jumps along J in w control the term in (8.2.7). For $d = 3$ we have that g is the restriction to a 2 dimensional object and we will use the Oswald interpolant as defined previously.

Suppose now we seek to approximate w_h, the dG approximation to (1.1.1) defined in Definition 2.2.7, by an element of V_{cdG} for some $\mathcal{T}_h$ decomposition. Consider $v_m \in V_{\text{cdG}}$ defined by

$$v_m := \begin{cases} \mathcal{O}s(w_h), & \text{on } \mathcal{T}_{\text{cG}}, \\ w_h, & \text{on } \mathcal{T}_{\text{dG}} \end{cases} \tag{8.2.9}$$

where $\mathcal{O}s(w_h)$ is that described in Corollary 8.2.6 with homogeneous Dirichlet boundary conditions on $\Gamma \cap \Gamma_{\text{cG}}$ and on J boundary conditions described by g above.

Lemma 8.2.10. *For a given $\mathcal{T}_h$ decomposition and dG approximation w_h, and with v_m defined in (8.2.9) (with g chosen as described above) we have*

$$|||w_h - v_m|||^2 \leq \Lambda_m \tag{8.2.11}$$

where

$$\Lambda_m = \sum_{e\in\mathcal{E}_{cG}} \Lambda_e \|[\![w_h]\!]\|^2_{L^2(e)} + \sum_{e\in J} \Lambda_e \|w_h^{\mathcal{C}} - g\|^2_{L^2(e)}$$

with

$$\Lambda_e := C_m(\varepsilon h_e^{-1} + h_e\|r^{1/2}\|^2_{L^\infty(\mathcal{T}_{cG})}) + \varepsilon\sigma h_e^{-1} + \frac{1}{2}\|b\|_{L^\infty(e)}$$

and C_m as defined in (8.2.7).

Proof. We examine each term in $|||w_h - v_m|||$ in turn using the properties of the

Oswald interpolant given in (8.2.7). This gives on cells

$$\sum_{E\in\mathcal{T}_h} \varepsilon|w_h - v_m|^2_{H^1(E)} = \sum_{E\in\mathcal{T}_{\mathrm{cG}}} \varepsilon|w_h - \mathcal{O}s(w_h)|^2_{H^1(E)}$$
$$\leq C_m \left(\sum_{e\in\mathcal{E}_{\mathrm{cG}}} \varepsilon h_e^{-1} \|[\![w_h]\!]\|^2_{L^2(e)} + \sum_{e\in J} \varepsilon h_e^{-1} \|w_h^{\mathcal{C}} - g\|^2_{L^2(e)} \right)$$

and

$$\sum_{E\in\mathcal{T}_h} \|r^{1/2}(w_h - v_m)\|^2_{L^2(E)}$$
$$\leq \|r^{1/2}\|^2_{L^\infty(\mathcal{T}_{\mathrm{cG}})} \sum_{E\in\mathcal{T}_{\mathrm{cG}}} \|w_h - \mathcal{O}s(w_h|_{\mathcal{T}_{\mathrm{cG}}})\|^2_{L^2(E)}$$
$$\leq C_m \left(\sum_{e\in\mathcal{E}_{\mathrm{cG}}} h_e \|r^{1/2}\|^2_{L^\infty(\mathcal{T}_{\mathrm{cG}})} \|[\![w_h]\!]\|^2_{L^2(e)} + \sum_{e\in J} h_e \|r^{1/2}\|^2_{L^\infty(\mathcal{T}_{\mathrm{cG}})} \|w_h^{\mathcal{C}} - g\|^2_{L^2(e)} \right).$$

Now $\mathcal{O}s(w_h)$ is continuous on edges in $\mathcal{E}_{\mathrm{cG}}$, i.e., $[\![v_m]\!] = 0$ for $e \in \mathcal{E}_{\mathrm{cG}}$, and $[\![w_h - v_m]\!] = 0$ for $e \in \mathcal{E}_{\mathrm{dG}} \setminus J$. On the discontinuous side of J we have $w_h^{\mathcal{D}} - v_m^{\mathcal{D}} = 0$ via the construction of v_m. Therefore for $e \in J$ we find $[\![w_h - v_m]\!] = (w_h^{\mathcal{C}} - v_m^{\mathcal{C}}) \cdot n^{\mathcal{C}} = (w_h^{\mathcal{C}} - g) \cdot n^{\mathcal{C}}$. So for the first edge terms we have

$$\sum_{e\in\mathcal{E}_h} \varepsilon\sigma h_e^{-1} \|[\![w_h - v_m]\!]\|^2_{L^2(e)} = \sum_{e\in\mathcal{E}_{\mathrm{cG}}} \varepsilon\sigma h_e^{-1} \|[\![w_h]\!]\|^2_{L^2(e)} + \sum_{e\in J} \varepsilon\sigma h_e^{-1} \|w_h^{\mathcal{C}} - g\|^2_{L^2(e)}$$

and for the second

$$\sum_{e\in\mathcal{E}_h} \frac{1}{2} \||b\cdot n|^{1/2} [\![w_h - v_m]\!]\|^2_{L^2(e)}$$
$$\leq \sum_{e\in\mathcal{E}_{\mathrm{cG}}} \frac{1}{2} \|b\|_{L^\infty(e)} \|[\![w_h]\!]\|^2_{L^2(e)} + \sum_{e\in J} \frac{1}{2} \|b\|_{L^\infty(e)} \|w_h^{\mathcal{C}} - g\|^2_{L^2(e)}.$$

Combining each of these results yields (8.2.11). □

Thus we have answered our first question: Given $\mathcal{T}_{\mathrm{cG}}$ we have formulated an upper bound on $\min |||w_h - v_m|||$ for a fixed problem. Now suppose we have an approximation w_h and wish to generate a cdG approximation using a $\mathcal{T}_h$ decomposition with the fewest degrees of freedom such that $|||w_h - v_h||| \leq (\Lambda_{\mathrm{ct}}/\Lambda_{\mathrm{cc}})\delta_m$ for

some δ_m using (8.2.2). We call δ_m the *tolerance*. Clearly if we pick δ_m too small then it may be that the only choice of $\mathcal{T}_h$ decomposition is $\mathcal{T}_h = \mathcal{T}_{\mathrm{dG}}$. The best way to illustrate the second question (in short, can we implement this method) is via numerical experiments.

8.3 Numerical Experiments

We briefly recap our notation: Λ_m is the difference between a dG approximation for a fixed problem and the Oswald interpolant for a given $\mathcal{T}_h$ decomposition; δ_m is the user chosen tolerance for a fixed problem which must be obtained through the choice of $\mathcal{T}_h$. We wish to show that we can determine (at least a close approximation to) the optimum decomposition automatically. We employ the following algorithm:

(1) Calculate the dG approximation w_h.

(2) Post process w_h, calculating Λ_m on each edge and for each cell, summing over the edges which belong to that cell.

(3) Set the value `tol`, the required bound on $|||w_h - v_h|||$.

(4) Order cells by $\Lambda_{\mathrm{ct}}/\Lambda_{\mathrm{cc}}$ multiplied by the value found in (2).

(5) Add cells to the continuous region from smallest first as determined by step (4) until the value of $\Lambda_{\mathrm{ct}}/\Lambda_{\mathrm{cc}} \sum_{e \in \mathcal{E}_{\mathrm{cG}}} \Lambda_m$ reaches `tol`.

This algorithm will be conservative, i.e., it may return a $\mathcal{T}_h$ decomposition which is not optimum due to the final step. As we know the value of $\Lambda_{\mathrm{ct}}/\Lambda_{\mathrm{cc}}$ (up to a constant) via Lemmas 8.1.1, 8.1.4 (or 8.1.8) we reasonably account for the sharpening of the layer in the choice of `tol`. We use `tol` and not δ_m to highlight the difference between the set and achieved value, i.e., by setting `tol` we will achieve $\delta_m <$ `tol`.

We present two examples with the aim of: Showing that the cdG method applies when the assumptions on the location of the boundary J made in Chapter 3 do not hold; demonstrating that the above algorithm and theory of the previous sections provides a practicable way of determining the $\mathcal{T}_h$ decomposition; and showing that the reduction in the degrees of freedom seen in Chapter 3 can be achieved when

the location of the layers is not know a priori. To these ends we present more complicated examples than previously.

Example 8.3.1 *Let $\Omega = (0,1)^2$. We seek to solve*

$$-\varepsilon\Delta u(x,y) + (1, 5\sin(10\pi x)) \cdot \nabla u(x,y) + u(x,y) = 1 \tag{8.3.1}$$

with homogeneous Dirichlet boundary conditions. This problem exhibits an exponential boundary layer at the outflow boundary $x = 1$ of width $\mathcal{O}(\varepsilon)$.

The streamlines defined by $b = (1, 5\sin(10\pi x))$ induce wave like behaviour in the approximation along the lines $y = 0$ and $y = 1$ as can be seen clearly in, e.g., Figure 8.3.4(b). The streamlines enter and leave the domain along the edges. It is not possible to determine a priori the hyperbolic approximation u_0 (in the notation of Chapter 3) or u_ε and therefore we cannot determine the $\mathcal{T}_h$ decomposition a priori using that approach. Even if we could determine these functions the construction of the continuous region would not be straightforward especially given Assumption 3.2.17 and the sinusoidal nature of b for this problem.

We set $\varepsilon = 10^{-3}$, $\sigma = 1$ and refine the domain uniformly into cells of edge length 2^{-5} and use piecewise bilinear elements. We have that $\|b\|_{L^\infty(E)}$ is bounded below by 1 and above by 5 and $\rho = 1$. It is straightforward to calculate Λ_m for each possible $\mathcal{T}_{\text{cG}}$.

In Figure 8.3.1 we plot the difference between the dG approximation w_h and cdG approximation v_h for decreasing tolerance/increasing degrees of freedom. The number of degrees of freedom for the fully discontinuous method is 4096 and for the continuous method 1089. For a relatively large tolerance (`tol` $= 1$) we have substantially fewer degrees of freedom but allow relatively large $\|w_h - v_h\|$ in the given norms. For a relatively small tolerance (`tol` $= 10^{-5}$) we have very close agreement between the approximations but do not save very many degrees of freedom over the continuous method. We do not see a uniform linear relationship between tolerance and $\|w_h - v_h\|$ as the algorithm is conservative, i.e., it always returns a $\mathcal{T}_h$ decomposition with a smaller error than `tol` and does not attempt to refine its estimate. Also note that some constants are unknown and so we do not recover

$|||w_h - v_h||| < \texttt{tol}$ but rather $|||w_h - v_h||| < C\texttt{tol}$, where C depends on C_m and the constant from the inverse inequality. This effect is more significant for smaller values of the tolerance.

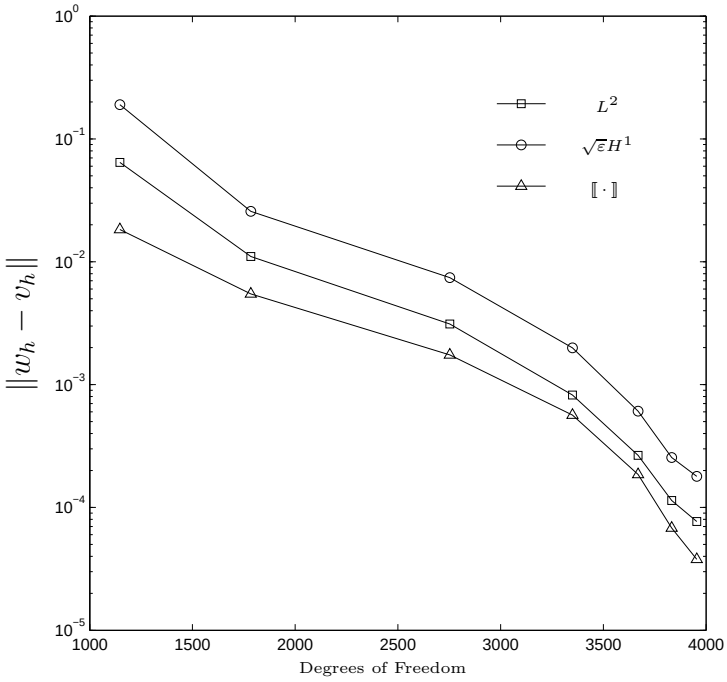
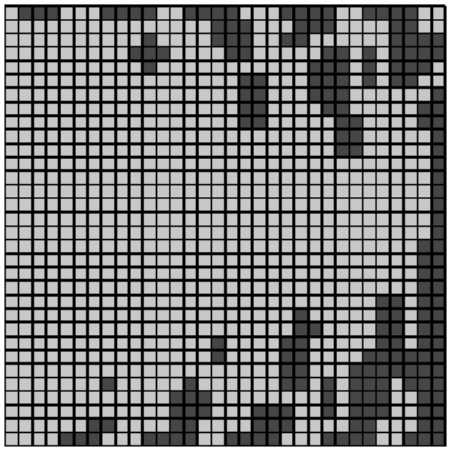


Figure 8.3.1: Example 8.3.1 with $\varepsilon = 10^{-3}$. The fully discontinuous method has 4096 degrees of freedom. From left to right $\texttt{tol} = 10^0, \ldots, 10^{-6}$.

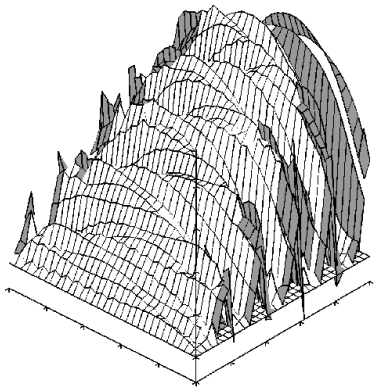
In Figures 8.3.2-8.3.4 we plot the cdG mesh and the cdG approximation for $\texttt{tol} = 10^{-1}, 10^{-3}$ and 10^{-5}. We see in Figure 8.3.2(a) that the $\mathcal{T}_{\text{cG}}$ region is large, but in Figure 8.3.2(b) we see slight non-physical oscillations entering the approximation. Figure 8.3.4(a) and (b) shows the opposite result in that we do not have any non-physical oscillations but we do not save a significant number of degrees of freedom. Figure 8.3.3 represents a compromise between the two extremes. Even with streamlines penetrating deeply into Ω from the boundary we can remove more than 20% of the degrees of freedom compared to the discontinuous method.

Example 8.3.2 *Let $\Omega = \{(x, y) : 1 \leq x^2 + y^2\}$. We seek to solve*

$$-\varepsilon\Delta u(x, y) + u_x(x, y) + \frac{1}{10}u(x, y) = 0 \tag{8.3.2}$$

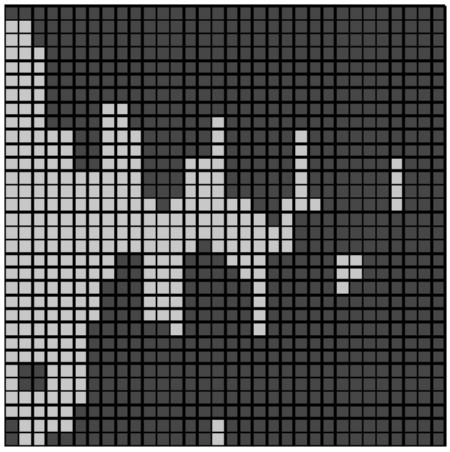


(a) $\mathcal{T}_h$ decomposition, $\mathcal{T}_{\mathrm{dG}}$ region darker.

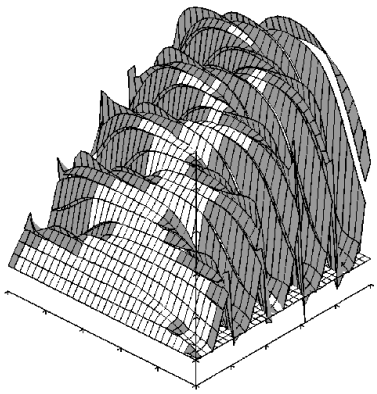


(b) cdG approximation, $\mathcal{T}_{\mathrm{dG}}$ region darker.

Figure 8.3.2: Example 8.3.1 with $\mathtt{tol} = 10^{-1}$ (1784 degrees of freedom).



(a) $\mathcal{T}_h$ decomposition, $\mathcal{T}_{\mathrm{dG}}$ region darker.

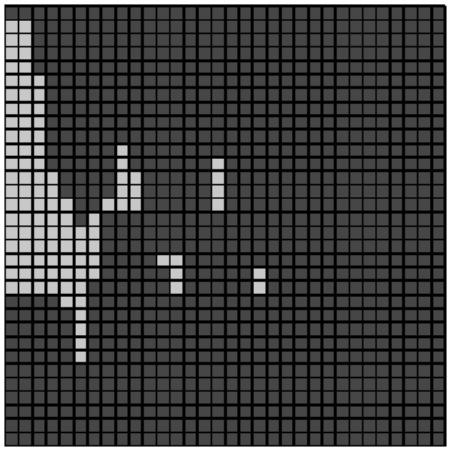


(b) cdG approximation, $\mathcal{T}_{\mathrm{dG}}$ region darker.

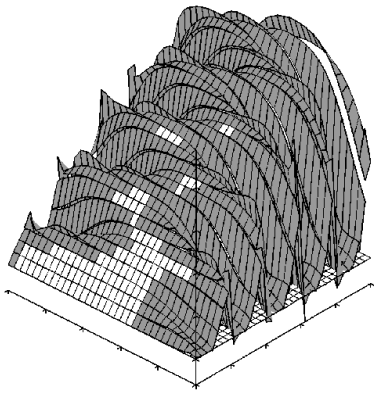
Figure 8.3.3: Example 8.3.1 with $\mathtt{tol} = 10^{-3}$ (3350 degrees of freedom).

with boundary conditions given by $u = 1$ *for* $x^2 + y^2 = 1$, *i.e., the inner boundary, and* $u \to 0$ *for* $x^2 + y^2 \to \infty$.

This is the example proposed as a model problem by Hemker [79] with some modifications. Firstly we have added a reaction term so $\rho \neq 0$. Secondly we approximate the problem as $x^2 + y^2 \to \infty$ by considering instead of the infinite plane the region $\Omega = \{(x, y) : 1 \leq x^2 + y^2, |x| \leq 10, |y| \leq 10\}$. The motivation of [79] was to find a problem more closely matching problems of industrial interest. The equation is simple but the more complex domain gives rise to an exponential layer at $\{x^2 + y^2 = 1, x \leq 0\}$ and characteristic layers extending



(a) $\mathcal{T}_h$ decomposition, $\mathcal{T}_{\mathrm{dG}}$ region darker.



(b) cdG approximation, $\mathcal{T}_{\mathrm{dG}}$ region darker.

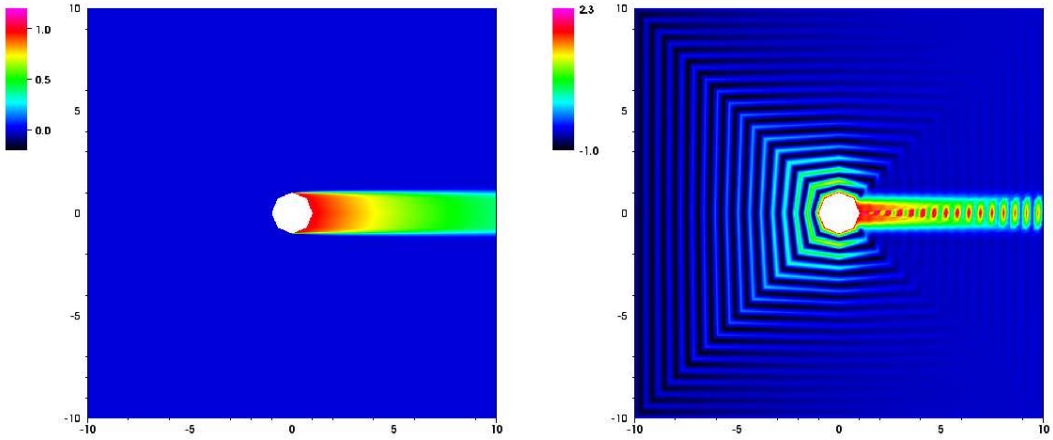
Figure 8.3.4: Example 8.3.1 with `tol` $= 10^{-5}$ (3833 degrees of freedom).

from the edge of the interior boundary. We will use the `deal.ii` standard mesh `hyper_cube_with_cylindrical_hole` to produce a 10×10 square domain with a radius 1 octagon removed from the centre (for further discussion of `deal.ii` see Chapter 9). Note that for the problem with $c = 0$ the analytic solution can be expressed in terms of modified Bessel functions.

We set $\varepsilon = 10^{-3}$ and $\sigma = 0.01$ and again use piecewise bilinear approximating polynomials. The mesh consists of 8192 quadrilaterals extending from the inner boundary. The cell size therefore varies with distance from the centre (hence we pick a smaller σ). Clearly $\|b\|_{L^\infty(\Omega)} = 1$ but the norm for each edge must be calculated on an edge by edge basis depending on its orientation.

In Figure 8.3.5(a) we plot a higher resolution dG approximation to the solution, i.e., on a more refined mesh using in excess of 2 million degrees of freedom compared to 32768 for the dG approximation on our standard mesh. The sharp layers at the front (i.e., $x \leq 0$) of the inner boundary and in the wake are apparent. In Figure 8.3.5(b) we plot the cG approximation on the standard mesh. The oscillations are clearly visible both in the wake and propagating against the flow forward of the inner boundary. Note that the scale is different in this plot and the oscillations extend far above and below the boundary.

As can be seen in Figure 8.3.5(a) there is a large region of the domain where the solution is almost identically zero and so we expect an approximation to have



(a) High resolution dG approximation. (b) cG appoximation.

Figure 8.3.5: Example 8.3.2 with $\varepsilon = 10^{-3}$. The oscillations in (b) are visible (note the scale).

small jumps there. We therefore expect to be able to make a substantial saving in degrees of freedom. We plot in Figure 8.3.6 the difference between the dG and cdG approximations in the given norms against the degrees of freedom returned for tolerances from 10^1 to 10^{-6}. As with Example 8.3.1 we see a decrease in the difference as the tolerance is decreased/degrees of freedom increased. Note now however the saving is much greater. Even for a conservative tolerance of $\texttt{tol} = 10^{-6}$ we only require 13852 degrees of freedom which is 42% of the number for a fully discontinuous approximation (or conversely 164% of the degrees of freedom required for the continuous method, compared to 388% for the discontinuous method).

In Figures 8.3.7-8.3.9 we plot the $\mathcal{T}_h$ decomposition and the approximations on $\mathcal{T}_{\text{cG}}$ and $\mathcal{T}_{\text{dG}}$ for $\texttt{tol} = 10^{-1}, 10^{-3}$ and 10^{-5}. We shade the $\mathcal{T}_{\text{dG}}$ region darker on the plots of the $\mathcal{T}_h$ decomposition*. We can see for $\texttt{tol} = 10^{-1}$ (Figure 8.3.7) the algorithm not only selects the regions far from the layer but also a portion between the characteristic layers. The region selected violates Assumption 3.2.17 from Chapter 3 as the streamlines are both inflow (to $\mathcal{T}_{\text{dG}}$) and outflow (from $\mathcal{T}_{\text{dG}}$) on J, the interface between $\mathcal{T}_{\text{cG}}$ and $\mathcal{T}_{\text{dG}}$. As the tolerance is decreased the $\mathcal{T}_{\text{cG}}$ region between the layers diminishes but the part of $\mathcal{T}_{\text{cG}}$ away from the layers remains large.

*It is not practical to plot the mesh as the thickness of the grid obscures the plot.

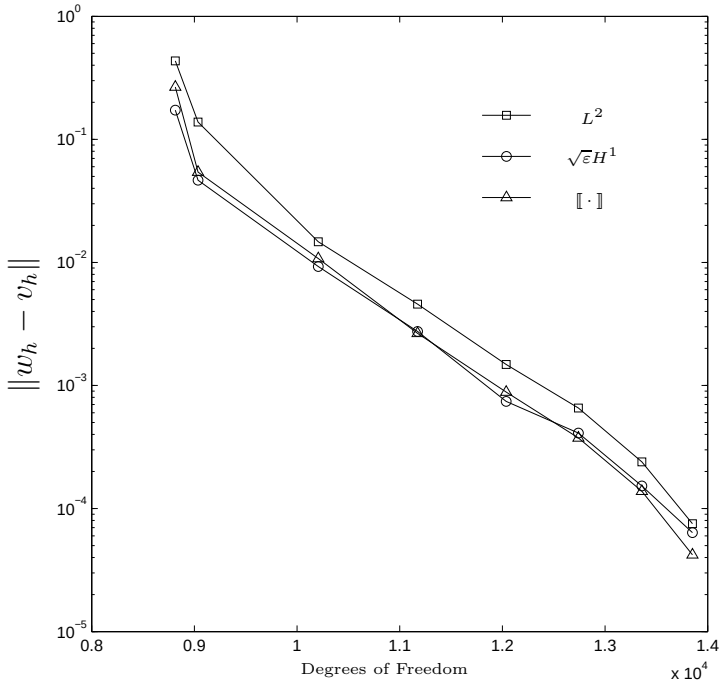
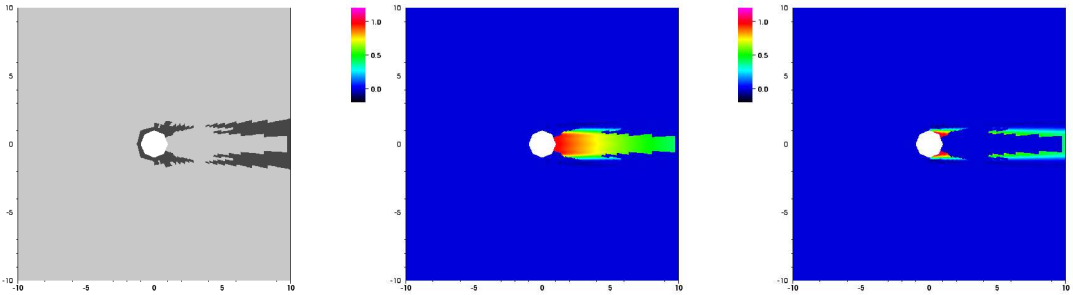


Figure 8.3.6: Example 8.3.2 with $\varepsilon = 10^{-3}$. The fully discontinuous method has 32768 degrees of freedom. From left to right $\texttt{tol} = 10^1, \ldots, 10^{-6}$.

Finally in Figure 8.3.10 we show a closer plot of the region round the inner boundary from Figure 8.3.9(c), the approximation on the $\mathcal{T}_{\text{dG}}$ region when $\texttt{tol} = 10^{-5}$. We can see that the discontinuous approximation still over or under shoots around the characteristic layers and boundary but these oscillations to not propagate outside of the discontinuous region.

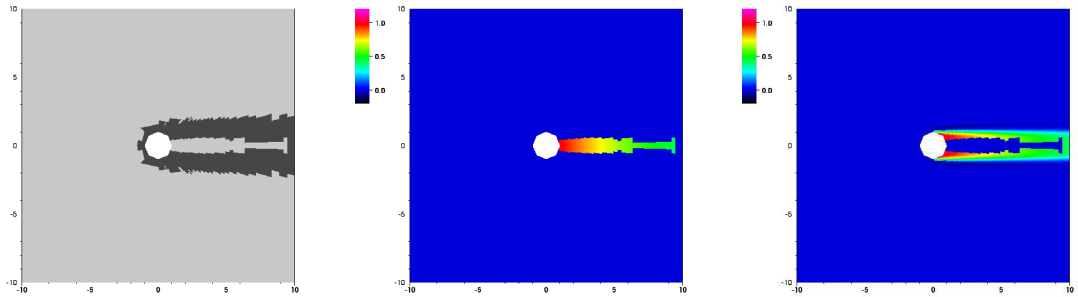


(a) $\mathcal{T}_h$ decomposition, $\mathcal{T}_{\text{dG}}$ region darker.

(b) Approximation on cG region.

(c) Approximation on dG region.

Figure 8.3.7: Example 8.3.2 with $\texttt{tol} = 10^{-1}$ (10208 degrees of freedom).

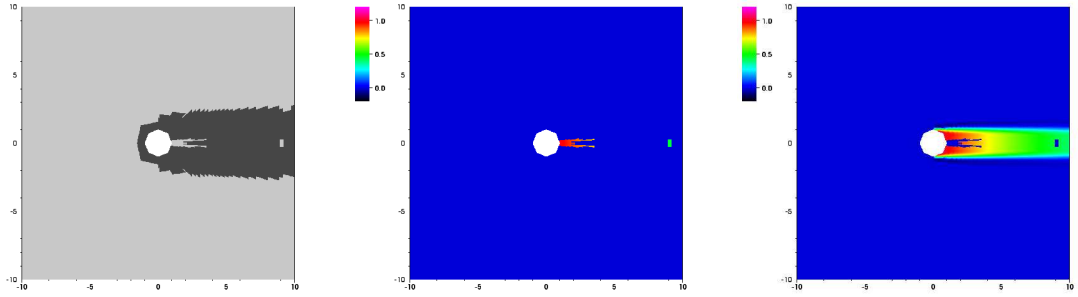


(a) $\mathcal{T}_h$ decomposition, $\mathcal{T}_{\mathrm{dG}}$ region darker.

(b) Approximation on cG region.

(c) Approximation on dG region.

Figure 8.3.8: Example 8.3.2 with $\texttt{tol} = 10^{-3}$ (12038 degrees of freedom).



(a) $\mathcal{T}_h$ decomposition, $\mathcal{T}_{\mathrm{dG}}$ region darker.

(b) Approximation on cG region.

(c) Approximation on dG region.

Figure 8.3.9: Example 8.3.2 with $\texttt{tol} = 10^{-5}$ (13359 degrees of freedom).

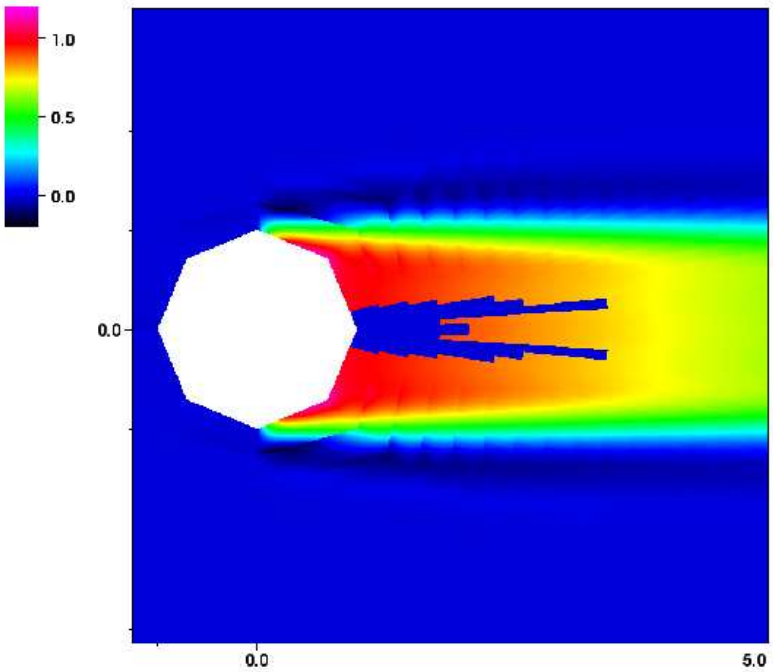


Figure 8.3.10: Detail from Example 8.3.2 with $\texttt{tol} = 10^{-5}$. It can be seen that the discontinuous approximation does extend above 1 and below 0.

Part IV

Implementation of the Continuous Discontinuous Galerkin Method

Chapter 9

Continuous Discontinuous Finite Element Code

In this chapter we present the code used to solve for the cdG method for the linear advection diffusion reaction problem. We first discuss the novel elements of the code before presenting an annotated code listing allowing a reader to implement this method.

9.1 A Note on Implementation in `deal.ii`

The work in this section has been previously published in part in [45]. We will discuss the implementation of the cdG method in `deal.ii`, a C++ finite element library. For more information about `deal.ii` see [1, 15, 16]. To make the code easier to read C++ keywords are typeset in **`Maroon`** and `deal.ii` classes are typeset in **`Navy Blue`**.

The cdG method poses several difficulties in implementation. In Chapter 7 (see also [46]) we prove that the dG approximation defined in Definition 2.2.7 tends to the cdG approximation defined in Definition 2.3.5 when the penalty parameter σ is increased on the $\mathcal{T}_{\text{cG}}$ part of the domain. This approach can be used to investigate the behaviour of the cdG method by modifying code which solves for the dG approximation. There are however clear drawbacks. The number of degrees of freedom (and hence the size of any matrices) will be equal in the cdG and dG methods using

this approach. Those entries in the system matrix corresponding to the super penalised jump terms may be very large as σ is large. This could lead to inefficiencies in numerical routines to invert the matrices. Thus when using the super penalty method we employ the direct solver provided by the deal.ii interface with the UMFPACK library.

The primary difficulty in the implementation of a cdG method in deal.ii (and other finite element libraries) is the lack of a native cdG element type. Without this element deal.ii cannot initialise a **Triangulation** with the correct number of degrees of freedom. Determining the number of degrees of freedom for the cdG method is a difficult task as the number of degrees of freedom per cell (or per edge for discontinuous elements) is determined by the type of shape function on the neighbouring element/edge. One option is to code a cdG element type and integrate it with deal.ii. However a more flexible approach is to use the existing capabilities of deal.ii to distribute the degrees of freedom by making use of the hp::**FECollection** capabilities of the library as described below. An advantage of this approach is that other capabilities of the library implemented for *hp* methods are automatically available.

We require in particular three properties of the deal.ii library. The first is the facility to group multiple finite elements into one data structure called a hp::**FECollection**. As the syntax suggests the usual use is for *hp* refinement to create a set of finite elements of the same type, e.g., scalar Lagrange elements **FE_Q**, but with different polynomial degrees, e.g., linear, quadratic. The second is the class **FESystem**. The purpose of this class is to create a vector valued finite element type. Finally we will use the special finite element type **FE_Nothing**. This is a finite element type with zero degrees of freedom. In Listing 9.1 we define **FESystem** and hp::**FECollection** objects and in Listing 9.2 we initialise those objects.

```
enum{CG_NOTHING = 0, NOTHING_DG = 1};
FESystem<dim>          c_fe;
FESystem<dim>          d_fe;
hp::FECollection<dim> fe_collection;
```

Listing 9.1: Declaration of hp::**FECollection**, **FESystem**

In Listing 9.2 c_fe is constructed as a two dimensional system of finite elements with linear Lagrange elements in the first vector location and a **FE_Nothing** element

in the second. The system d_fe is set up in opposition with FE_Nothing in the first space and linear discontinuous shape functions in the second. This construction is to avoid human coding error at a later stage as selecting the incorrect entry will produce an obvious effect. These two systems are then inserted into fe_collection.

```
c_fe (FE_Q<dim>(1), 1, FE_Nothing<dim>(), 1);
d_fe (FE_Nothing<dim>(), 1, FE_DGQ<dim>(1), 1);
fe_collection.push_back (c_fe);
fe_collection.push_back (d_fe);
```

Listing 9.2: Initialisation of hp::FECollection, FESystem

On a Triangulation each cell is flagged as being either in the continuous or discontinuous region (we have assumed that the regions align with the mesh as with the $\mathcal{T}_h$ decomposition) using material_id. The active_fe_index is then set on each cell as shown in Listing 9.3. As FE_Nothing has zero degrees of freedom the total number of degrees of freedom for the method will be correct. Only one finite element with degrees of freedom is active on each cell.

```
for (typename hp::DoFHandler<dim>::active_cell_iterator
      cell = dof_handler.begin_active();
      cell != dof_handler.end(); ++cell)
{
  if (cell_is_c(cell))
    cell->set_active_fe_index (CG_NOTHING);
  else if (cell_is_d(cell))
    cell->set_active_fe_index (NOTHING_DG);
  else
    // ...throw exception...
}
```

Listing 9.3: Setting the correct active_fe_index

Now when we call dof_handler.distribute_dofs(fe_collection) the degrees of freedom will be correctly distributed to the triangulation. We could now simply initialise a SparsityPattern with this dof_handler but we can save some memory by assigning the coupling permitted between finite element types. Then calling make_flux_sparsity_pattern does not assume coupling between every element in the sparsity pattern.

Once we have initialised the relevant matrices we may now proceed to assembly. It is important that on each cell the correct entry from fe_collection is used, and in turn the correct element of FESystem on that element. In particular edges in J, the boundary between $\mathcal{T}_{\mathrm{cG}}$ and $\mathcal{T}_{\mathrm{dG}}$, have to be carefully considered. When

assembling the jump between a dG and cG element on J it is not possible to initialise an **FEFaceValues** object on the cG element. This is of course perfectly natural as continuous Lagrange finite elements do not have support (in the deal.ii sense) on the edges of cells. To work around this difficulty we must extract the locations of the support points on the face and then calculate their values on the neighbouring cG cell.

An interesting consequence of this approach is that we create two solutions: one for the c_fe component and another for the d_fe component. We present a simple example to demonstrate this but remark that the effect can also be seen in Figures 8.3.2-8.3.4.

Example 9.1.1 *Let $\Omega = (0,1)^2$. We seek to solve*

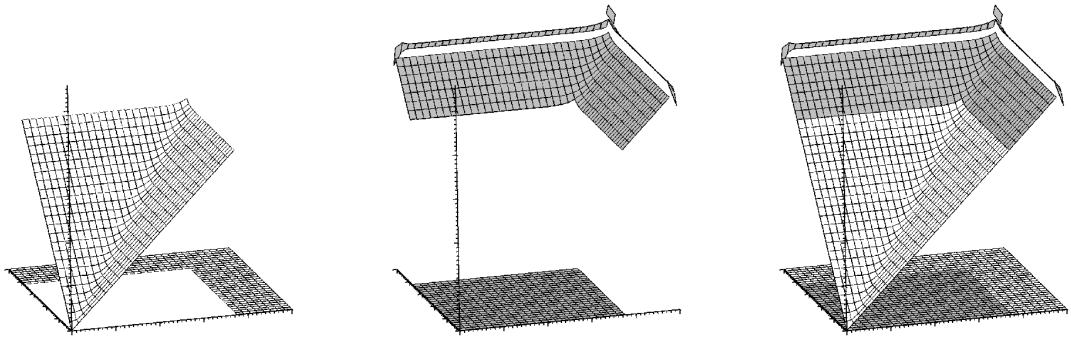
$$-\varepsilon\Delta u(x,y) + (1,1)\cdot\nabla u(x,y) = 1$$

with boundary conditions given by

$$u(x,y) = x + y(1-x) + \frac{\mathrm{e}^{-1/\varepsilon} - \mathrm{e}^{-(1-x)(1-y)/\varepsilon}}{1-\mathrm{e}^{-1/\varepsilon}}.$$

This problem exhibits an exponential boundary layer at the outflow boundaries $x = 1$ and $y = 1$ of width $\mathcal{O}(\varepsilon)$.

Note that this example does not conform to the usual requirement that $\rho > 0$. We set $\varepsilon = 10^{-3}$ and calculate the cdG and dG approximations on a uniform 32×32 grid of squares. We select $\mathcal{T}_{\mathrm{cG}} = [0, 0.75]^2$. This is not optimum but our aim is to illustrate the unusual appearance of approximations using **FE_Nothing**. In Figure 9.1.1 we plot the cdG approximation and the dG-**FE_Nothing** and **FE_Nothing**-cG components of the cdG approximation. The approximation on $\mathcal{T}_{\mathrm{dG}}$ is shaded in each case. The presence of the **FE_Nothing** component is apparent.



(a) The cG-**FE_Nothing** component of the cdG approximation.

(b) The **FE_Nothing**-dG component of the cdG approximation.

(c) cdG approximation.

Figure 9.1.1: Example 9.1.1 with $\varepsilon = 10^{-3}$. The presence of the **FE_Nothing** component is clear. The approximation on $\mathcal{T}_{\mathrm{dG}}$ in each case is shaded.

9.2 Commented Code

The code for cdG and incompressible miscible displacement runs to over 4000 lines (excluding comments) and so inclusion of the complete code is not practical. In Section 9.1 we have already explained the novel parts of the cdG code. Here we introduce enough material to allow the reader to understand the important features of `deal.ii` and in particular to understand and recreate the implementation of the cdG method. The code presented generates an approximation to

$$\begin{aligned} -\varepsilon\Delta u + b\cdot\nabla u + cu &= f \qquad \text{in } \Omega, \\ u &= g \qquad \text{on } \partial\Omega \end{aligned}$$

using the following interior penalty bilinear and linear forms:

$$\mathcal{B}_\varepsilon(u,v) = \mathcal{B}_d(u,v) + \mathcal{B}_{ar}(u,v) = \ell(f,g;v) \tag{9.2.2}$$

where

$$\begin{aligned} \mathcal{B}_d(u,v) = &\sum_{E\in\mathcal{T}_h}\int_E \varepsilon\nabla u\cdot\nabla v\,\mathrm{d}\boldsymbol{x} \\ &+\sum_{e\in\mathcal{E}_h}\int_e \sigma\frac{\varepsilon}{h_e}[\![u]\!]\cdot[\![v]\!] - (\varepsilon\{\!\{\nabla u\}\!\}\cdot[\![v]\!] + \vartheta\varepsilon\{\!\{\nabla v\}\!\}\cdot[\![u]\!])\,\mathrm{d}s, \end{aligned}$$

$$\mathcal{B}_{ar}(u,v) = \sum_{E\in\mathcal{T}_h}\int_E -(b\cdot\nabla v)u - (\nabla\cdot b)uv + cuv\,\mathrm{d}\boldsymbol{x} + \sum_{e\in\mathcal{E}_h^o}\int_e b\cdot[\![v]\!]u^{\in}\,\mathrm{d}s + \sum_{e\in\Gamma^{\text{out}}}\int_e (b\cdot n)uv\,\mathrm{d}s$$

and

$$\ell(f,g;v) = \sum_{E\in\mathcal{T}_h}\int_E fv\,\mathrm{d}\boldsymbol{x} + \sum_{e\in\Gamma}\int_e\left(\sigma\frac{\varepsilon}{h_e}v - \vartheta\varepsilon\nabla\cdot v\right)g\,\mathrm{d}s - \sum_{e\in\Gamma^{\text{in}}}\int_e (b\cdot n)vg\,\mathrm{d}s.$$

The super penalty approach of Chapter 7, the decoupled and weighted forms of Chapter 3, or using a file to generate the $\mathcal{T}_h$ decomposition a priori can be implemented with some minor modifications. For incompressible miscible displacement our implementation is close to that of the `deal.ii` tutorial Step-21 [1] and we direct readers there for a thorough discussion of the use of Raviart-Thomas elements, time dependent problems and solution using the Schur Complement.

We do not include all detail from class declarations and function protoypes as our purpose is not to describe how to write a program in C++ but how to write one in `deal.ii`. It suffices to say that each described function and class must have somewhere a prototype. The code is written favouring readability over efficiency. This differs from professionally written code (in general) where efficiency takes a prominent role. The code here is *templated* by dimension as it was written to make expansion to higher dimensions possible. However many of the routines presented here assume two dimensions as the dimension independent generalizations add complexity without adding any insight for the reader.

Header files

The library is organized in multiple sections and is is necessary to include the following header files, along with some headers from the Standard Template Library (STL) allowing us access to some common C++ routines.

```
#include <deal.II/base/quadrature_lib.h>
#include <deal.II/base/logstream.h>
#include <deal.II/base/function.h>
#include <deal.II/base/utilities.h>
```

```
#include <deal.II/base/parameter_handler.h>
#include <deal.II/base/parsed_function.h>
#include <deal.II/base/table_handler.h>
#include <deal.II/base/timer.h>

#include <deal.II/lac/vector.h>
#include <deal.II/lac/full_matrix.h>
#include <deal.II/lac/sparse_matrix.h>
#include <deal.II/lac/sparse_direct.h>
#include <deal.II/lac/constraint_matrix.h>
#include <deal.II/lac/precondition_block.h>

#include <deal.II/grid/tria.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/tria_accessor.h>
#include <deal.II/grid/tria_iterator.h>
#include <deal.II/grid/grid_refinement.h>
#include <deal.II/grid/grid_tools.h>
#include <deal.II/grid/grid_out.h>

#include <deal.II/dofs/dof_tools.h>
#include <deal.II/dofs/dof_accessor.h>

#include <deal.II/fe/fe_q.h>
#include <deal.II/fe/fe_dgq.h>
#include <deal.II/fe/mapping_q1.h>
#include <deal.II/fe/fe_nothing.h>
#include <deal.II/fe/fe_system.h>
#include <deal.II/fe/fe_values.h>
#include <deal.II/fe/fe_tools.h>

#include <deal.II/hp/dof_handler.h>
#include <deal.II/hp/fe_collection.h>
#include <deal.II/hp/fe_values.h>

#include <deal.II/numerics/vectors.h>
#include <deal.II/numerics/data_out.h>
#include <deal.II/numerics/error_estimator.h>
#include <deal.II/numerics/matrices.h>
#include <deal.II/numerics/fe_field_function.h>

#include <fstream>
#include <sstream>
#include <iostream>
#include <map>
#include <boost/lexical_cast.hpp>
```

To simplify the code we use the `deal.ii` namespace

```
using namespace dealii;
```

and so we need not prepend each function from the library with "dealii.".

Class: ParameterReader

We do not want to recompile the code to make simple changes to the parameters. Fortunately the deal.ii library includes a class to interface with a parameter file, and provides an interface via ParsedFunction to the (non-deal.ii) fparser class. An example of a parameter file can be found in Section 9.3.

The inheritance of the Subscriptor class prevents the destruction of the ParameterHandler object passed if the ParameterReader object still exists. Here this is a sensible safety precaution as the ParameterHandler is passed by reference and so its destruction before the ParameterReader object would break the reference and most likely cause a segmentation fault.

```
class ParameterReader : public Subscriptor
{
  public:
    ParameterReader(ParameterHandler &);
    ~ParameterReader () {};

    void read_parameters(const std::string);

  private:
    void declare_parameters();
    ParameterHandler &prm;
};
```

The constructor simply takes the given ParameterHandler and copies its attributes (by reference) into an internal handler prm.

```
ParameterReader::ParameterReader(ParameterHandler &paramhandler)
                :
                prm(paramhandler) {}
```

The parameters are grouped into subsections. On entering a subsection each entry is declared by giving a name, default value, type (e.g., Patterns::Double (0) declares a positive double) and short description. In the parameter file the parameters must likewise be grouped into subsections but need not appear in the same order. If a parameter is not declared it will take the default value and a warning will be output to the console at run time. The variable names are self explanatory,

except for `beta` which is the advection coefficient b, chosen so to avoid single letter variable names.

```
void ParameterReader::declare_parameters()
{
  prm.enter_subsection ("Equation Data");
  {
    prm.declare_entry ("epsilon", "0.001",Patterns::Double(0),
                       "Diffusion coefficient");
    prm.declare_entry("theta", "1",
                      Patterns::Integer(),
                      "Switch between Interior Penalty types");
    prm.declare_entry ("sigma", "10.0",Patterns::Double(0),
                       "Penalty parameter");
    prm.declare_entry ("xmin","0.0",Patterns::Double(),
                       "Domain minimum x");
    prm.declare_entry ("xmax","1.0",Patterns::Double(),
                       "Domain maximum x");
    prm.declare_entry ("ymin","0.0",Patterns::Double(),
                       "Domain minimum y");
    prm.declare_entry ("ymax","1.0",Patterns::Double(),
                       "Domain maximum y");
  }
  prm.leave_subsection ();
  prm.enter_subsection ("Run Options");
  {
    prm.declare_entry("true present", "true",
                  Patterns::Bool(),
                  "Is the true solution present?");
    prm.declare_entry("print parameters", "true",
                  Patterns::Bool(),
                  "print parameters at the start of the run?");
    prm.declare_entry("L2 jump tolerance", "0.0001",
                  Patterns::Double(0),
                  "L2 jump tolerance of dG solution "
                  "when refining grid for cdG");
    prm.declare_entry("initial refinement", "3",
                  Patterns::Integer(0),
                  "Number of refinements of basic grid");
    prm.declare_entry("refinement steps", "1",
                  Patterns::Integer(0),
                  "Number of refinement iterations");
    prm.declare_entry("Fast", "true",
                  Patterns::Bool(),
                  "If true do not calculate the dG norms");
    prm.declare_entry("skipcdGNorm", "false",
                  Patterns::Bool(),
                  "If true do not calculate the cdG norms");
  }
```

```
  prm.leave_subsection();
```

For functions the form is different. Each function is initialized as a **ParsedFunction** object and at this point any constants and variables are read from the file. However the functions are only *declared* (with a default value, here $x + y$). They cannot be used until initialized using `parse_parameters`. The second argument in `declare_parameters` sets the number of components of the function (the default being 1, a scalar function).

```
  prm.enter_subsection ("Beta Data");
  {
    Functions::ParsedFunction<dim>::declare_parameters(prm,2);
    prm.set("Function expression","1; 1");
  }
  prm.leave_subsection ();
  prm.enter_subsection("divBeta Data");
  {
    Functions::ParsedFunction<dim>::declare_parameters(prm);
    prm.set("Function expression", "x+y");
  }
  prm.leave_subsection();
  prm.enter_subsection("True solution");
  {
    Functions::ParsedFunction<dim>::declare_parameters(prm);
    prm.set("Function expression", "x+y");
  }
  prm.leave_subsection();
  prm.enter_subsection ("Right Hand Side Data");
  {
    Functions::ParsedFunction<dim>::declare_parameters(prm);
    prm.set("Function expression", "x+y");
  }
  prm.leave_subsection();
  prm.enter_subsection ("Boundary Data");
  {
    Functions::ParsedFunction<dim>::declare_parameters(prm);
    prm.set("Function expression", "x+y");
  }
  prm.leave_subsection();
  prm.enter_subsection ("Reaction Data");
  {
    Functions::ParsedFunction<dim>::declare_parameters(prm);
    prm.set ("Function expression", "x+y");
  }
  prm.leave_subsection();
}
```

The parameters are read (and outputted, depending on the value of `print parameters`) in the final code of this subsection.

```
void ParameterReader::read_parameters(
                                const std::string parameter_file)
{
  declare_parameters();
  std::cout << "   Reading parameter file: "
            << parameter_file
            << std::endl;
  prm.read_input (parameter_file);

  prm.enter_subsection ("Run Options");
    bool print = prm.get_bool("print parameters");
  prm.leave_subsection();
  if(print)
  {
    prm.print_parameters (std::cout, ParameterHandler::Text);
    std::cout <<
          "# ***************END OF PARAMETERS***************"
          <<std::endl;
  }
}
```

Class: cdGMethod

The main body of the code consists of those routines for setting up the problem, managing its execution and outputting any results.

In the class declaration we have shortened the function prototypes but leave the variable declarations.

```
template <int dim>
class cdGMethod
{
  public:
    cdGMethod (ParameterHandler &, unsigned int);
    ~cdGMethod ();

    void run ();

  private:
    const static unsigned char c_boundary_id = 'b';
    const static unsigned char c_domain_id = 'c';
    const static unsigned char d_domain_id = 'd';
    enum {CG_NOTHING = 0, NOTHING_DG = 1};
```

```
    //...function prototypes omitted...

    ParameterHandler      &prm;
    Triangulation<dim>    triangulation;
    const MappingQ1<dim>  mapping;
    hp::MappingCollection<dim> mapping_collection;
    FESystem<dim>         c_fe, d_fe;
    hp::FECollection<dim> fe_collection;
    hp::DoFHandler<dim>   dG_dof_handler, cdG_dof_handler;
    hp::QCollection<dim>  q_collection;
    const QGauss<dim-1>   face_quadrature;
    ConstraintMatrix      constraints;
    SparsityPattern       sparsity_pattern;
    SparseMatrix<double>  system_matrix;
    const ADEquation<dim> ad_equation;
    Vector<double>        dG_solution, cdG_solution, system_rhs;
    Functions::ParsedFunction<dim> true_solution;
    Functions::ParsedFunction<dim> boundary_values;
    double                dG_time, cdG_time;
    bool                  true_present;
    bool                  cdG_run;
    unsigned int          run_number;
    unsigned int          refinement;

    //...output variables omitted...
};
```

The constructor first uses an initializer list to set up what variables it can.

```
template <int dim>
cdGMethod<dim>::cdGMethod (ParameterHandler &param,
                           unsigned int refstep)
  :
  prm(param),
```

The triangulation details the mesh and nodes. The option `maximum_smoothing` ensures that any hanging nodes present conform to `deal.ii` internal requirements. The mapping is a default constructor of the **`MappingQ1`** class for straight line mappings between the unit and general cell.

```
  triangulation (Triangulation<dim>::maximum_smoothing),
  mapping(),
```

The construction of two mismatched **`FESystem`** has been described in Section 9.1. Here we pick linear continuous (**`FE_Q`**) and discontinuous (**`FE_DGQ`**) elements. Two handlers for the degrees of freedom are required in the case of a $\mathcal{T}_h$ decomposition.

```
  c_fe (FE_Q<dim>(1),1,
        FE_Nothing<dim>(),1),
```

```
  d_fe (FE_Nothing<dim>(), 1,
        FE_DGQ<dim>(1), 1),
  dG_dof_handler (triangulation),
  cdG_dof_handler (triangulation),
  face_quadrature (2),
```

The object `ad_equation` governs the parameters of the advection diffusion reaction equation. In principle changing this object will result in cdG methods for other equations. The `true_solution` and `boundary_values` are initialized as length 2 vectors of scalar functions as declared in **`ParameterHandler`**. This is because we have two approximations, two degree of freedom handlers, etc., corresponding to the approximation on the continuous and discontinuous regions as described in Section 9.1.

```
  ad_equation (param),
  true_solution(2),
  boundary_values(2)
{
```

Each of the continuous and discontinuous regions have their own quadrature formula, fixed here to two points in each spatial dimension (exact for piecewise linear functions).

```
  const QGauss<dim> c_quadrature(2);
  const QGauss<dim> d_quadrature(2);
```

The `fe_collection`, `q_collection` and `mapping_collection` behave like a vectors taking taking the finite element systems, quadrature formula and mapping for the continuous and discontinuous elements respectively.

```
  fe_collection.push_back (c_fe);
  q_collection.push_back (c_quadrature);
  mapping_collection.push_back (mapping);

  fe_collection.push_back (d_fe);
  q_collection.push_back (d_quadrature);
  mapping_collection.push_back (mapping);
```

Finally we look into the parameter file to see if any mesh or true solution has been specified and store the data. If the true solution is present it must be parsed by interfacing with the **`ParsedFunction`** class, which in turn interfaces with the (non-`deal.ii`) `fparser` library. See the parameter file for more details of how to declare functions in this way, but note that as we have initialized `true_solution`

as having two scalar components the parameter file must reflect this. Somewhat confusingly this makes the declaration in the parameter file the same for a vector valued function, e.g., the advection coefficient, and two scalar valued functions, e.g., `true_solution`. The variable `refinement` will be used to refine the initial triangulation, i.e., determine the mesh size.

```
  prm.enter_subsection("Run Options");
    true_present = prm.get_bool("true present");
  prm.leave_subsection();

  if(true_present)
  {
    prm.enter_subsection("True solution");
      true_solution.parse_parameters(prm);
    prm.leave_subsection();
  }
  prm.enter_subsection("Boundary Data");
    boundary_values.parse_parameters(prm);
  prm.leave_subsection();
  prm.enter_subsection ("Run Options");
    const int initrefinement
                        = prm.get_integer("initial refinement");
  prm.leave_subsection ();
  refinement = initrefinement + refstep;
}
```

We also need a destructor, which need not do anything as it will only be called as the program completes. However to conform to good memory management we release any memory allocated to the (potentially very large) degree of freedom handlers.

```
template <int dim>
cdGMethod<dim>::~cdGMethod ()
{
  dG_dof_handler.clear ();
  cdG_dof_handler.clear ();
}
```

Initializing `triangulation` in the constructor has only set up the internals. We need to be able to associate a specific grid with this triangulation. We get the extent of the grid from the parameter file (here fixed to two dimensions) and then use the `deal.ii` **`GridGenerator`** to make a rectangle which is uniformly refined a number of times as determined by the value of `refinement`.

```
template <int dim>
```

```
void cdGMethod<dim>::make_grid ()
{
  prm.enter_subsection ("Equation Data");
    const double xmin = prm.get_double("xmin");
    const double ymin = prm.get_double("ymin");
    const double xmax = prm.get_double("xmax");
    const double ymax = prm.get_double("ymax");
  prm.leave_subsection ();

  const Point<dim> min(xmin,ymin);
  const Point<dim> max(xmax,ymax);

  GridGenerator::hyper_rectangle (triangulation,min,max);
  triangulation.refine_global (refinement);
}
```

The code is written so that it always generates two approximations. Regardless of any settings it produces a dG approximation. This is required in order to generate the $\mathcal{T}_h$ decomposition. The next function manages the $\mathcal{T}_h$ decomposition of the mesh.

```
template <int dim>
void cdGMethod<dim>::modify_grid ()
{
  if(cdG_run)
  {
    std::cout << "Trying to generate a mesh based on dG"
              << std::endl;
```

The **Assert** class is an effective way of tracking potential errors. If the assertion is triggered the program terminates at the **Assert** in a controlled manner and outputs to the console the nature of the exception.

```
    Assert(dG_solution.size() != 0,
           ExcMessage("You have tried to start a cdG run but no
               dG approximation exists"));
    Vector<double> dummy;
```

We calculate the size of the jumps at interelement boundaries in order to decompose the mesh.

```
    ad_equation.calculate_L2_jump_norms (dG_dof_handler,
                                         q_collection,
                                         mapping_collection,
                                         face_quadrature,
                                         dG_solution,
                                         false,
                                         L2_jump_norm,
                                         dummy);
    prm.enter_subsection("Run Options");
```

```
    const double tol = prm.get_double("L2 jump tolerance");
  prm.leave_subsection();
```

This is the first appearance of a loop over every cell. The `Triangulation` class enables iterators to every cell (or active cell, i.e., those at the lowest refinement level). To access faces we prefer to loop over every cell and then visit the faces of that cell in turn by accessing `cell->face(number)`, although iterators for the faces do exist. Here if every jump at the faces of an element is small we mark that element as in the continuous region and use the `user_flag` attribute of the face to store this. Cells are marked using the `material_id` flag, which is more flexible but is not implemented for faces.

```
  typename Triangulation<dim>::active_cell_iterator
    cell = triangulation.begin_active(),
    endc = triangulation.end();
  for (; cell!=endc; ++cell)
  {
    unsigned int face_count = 0;
    for (unsigned int face_no=0;
         face_no<GeometryInfo<dim>::faces_per_cell;
         ++face_no)
    {
      if(L2_jump_norm(cell->face(face_no)->index()) < tol)
        ++face_count;
      else
        cell->face(face_no)->set_user_flag();
    }

    if(face_count == GeometryInfo<dim>::faces_per_cell)
    {
      cell->set_material_id(c_domain_id);
      if (cell->at_boundary())
        for (unsigned int f=0;
              f<GeometryInfo<dim>::faces_per_cell;
              ++f)
          if (cell->face(f)->at_boundary())
            cell->face(f)->set_all_boundary_indicators(
               c_boundary_id);
    }
    else
    {
      cell->set_material_id(d_domain_id);
      for (unsigned int face_no=0;
            face_no<GeometryInfo<dim>::faces_per_cell;
            ++face_no)
        cell->face(face_no)->set_user_flag();
```

```
      }
    }
  }
```

Now we cover the initial dG approximation by setting the entire domain to be in the discontinuous region.

```
  else if (!cdG_run)
  {
    for (typename Triangulation<dim>::active_cell_iterator
          cell = triangulation.begin_active();
          cell != triangulation.end(); ++cell)
      cell->set_material_id(d_domain_id);
  }
```

Finally in the case of a cdG approximation check that we do not have any isolated continuous cells.

```
  if(cdG_run)
  {
    typename Triangulation<dim>::active_cell_iterator
      cell = triangulation.begin_active(),
      endc = triangulation.end();
    for (; cell!=endc; ++cell)
    {
      if(cell->material_id() == d_domain_id) continue;
      else
      {
        unsigned int count = 0;
        for (unsigned int face_no=0;
              face_no<GeometryInfo<dim>::faces_per_cell;
              ++face_no)
        {
          if (cell->face(face_no)->at_boundary()) ++count;
          else if (cell->neighbor(face_no)->material_id()
                      == d_domain_id) ++count;
        }
        if (count == GeometryInfo<dim>::faces_per_cell)
          cell->set_material_id(d_domain_id);
      }
    }
  }
}
```

The next two routines allow access to the `material_id` of a cell in safety, i.e., without the risk of changing the `material_id`.

```
template <int dim>
bool cdGMethod<dim>::cell_is_c
  (const typename hp::DoFHandler<dim>::cell_iterator &cell)
```

```
{
  return (cell->material_id() == c_domain_id);
}
template <int dim>
bool cdGMethod<dim>::cell_is_d
  (const typename hp::DoFHandler<dim>::cell_iterator &cell)
{
  return (cell->material_id() == d_domain_id);
}
```

Once we have marked each cell as either in the continuous or discontinuous region we set the `active_fe_index` on the mesh so that the correct element of `fe_collection` we be active on the cell (using the enumeration from the class declaration).

```
template <int dim>
void cdGMethod<dim>::set_active_fe_indices (
                                hp::DoFHandler<dim> &dof_handler)
{
  for (typename hp::DoFHandler<dim>::active_cell_iterator
        cell = dof_handler.begin_active();
        cell != dof_handler.end(); ++cell)
  {
    if (cell_is_c(cell))
      cell->set_active_fe_index (CG_NOTHING);
    else if (cell_is_d(cell))
      cell->set_active_fe_index (NOTHING_DG);
    else
      Assert (false, ExcMessage("Unexpected cell type"));
  }
}
```

One of the major advantages to using a library such as `deal.ii` to generate code is the existence of routines to perform complicated tasks efficiently. Here we wish to create a **`SparsityPattern`** for the problem representing the non-zero entries in a sparse matrix and initialize our `system_matrix` and `system_rhs` (that is, the mass matrix A and right hand side vector f representing the discrete problem $Ax = f$, where x is the dG or cdG approximation we are calculating). We use a `CompressedSimpleSparsityPattern` for the purposes of memory management and as our problem has face (flux) coupling `make_flux_sparsity_pattern`. We must also instruct the sparsity pattern that on the different regions (and on the interface) we have different coupling, which we do using **`DoFTools`**`::Coupling`.

```
template <int dim>
```

```
void cdGMethod<dim>::setup_dofs (
                                hp::DoFHandler<dim> &dof_handler,
                                Vector<double> &solution)
{
```

First inform the `dof_handler` of our active indices and then calculate the number of degrees of freedom (recall `FENothing` contributes no degrees of freedom).

```
  set_active_fe_indices (dof_handler);
  dof_handler.distribute_dofs (fe_collection);
```

We have no constraints here but the `ConstraintMatrix` class is an efficient way to handle the problem during assembly, so we make an empty `constraints` then continue as described above.

```
  constraints.clear ();
  constraints.close ();
  CompressedSimpleSparsityPattern csp (dof_handler.n_dofs(),
                                       dof_handler.n_dofs());
  Table<2,DoFTools::Coupling>
                  cell_coupling (fe_collection.n_components(),
                                 fe_collection.n_components());
  Table<2,DoFTools::Coupling>
                  face_coupling (fe_collection.n_components(),
                                 fe_collection.n_components());
  cell_coupling[NOTHING_DG][NOTHING_DG] = DoFTools::none;
  face_coupling[NOTHING_DG][NOTHING_DG] = DoFTools::always;
  cell_coupling[CG_NOTHING][CG_NOTHING] = DoFTools::always;
  face_coupling[CG_NOTHING][CG_NOTHING] = DoFTools::none;
  cell_coupling[CG_NOTHING][NOTHING_DG] = DoFTools::none;
  face_coupling[CG_NOTHING][NOTHING_DG] = DoFTools::always;
  cell_coupling[NOTHING_DG][CG_NOTHING] = DoFTools::none;
  face_coupling[NOTHING_DG][CG_NOTHING] = DoFTools::always;

  DoFTools::make_flux_sparsity_pattern (dof_handler, csp,
                                        cell_coupling,
                                        face_coupling);
  constraints.condense (csp);
  sparsity_pattern.copy_from (csp);
```

Then we can resize the matrix and right hand side vector, as well as the dG or cdG approximation `solution` depending on the type of approximation we are performing.

```
  system_matrix.reinit (sparsity_pattern);
  solution.reinit (dof_handler.n_dofs());
  system_rhs.reinit (dof_handler.n_dofs());
}
```

Now we come to the logical assembly of the mass matrix and right hand side. This function interfaces with the `ADEquation` class where the actual calculation of the weights for each entry in the matrix is performed. This construction allows for a different equation or method to be interchanged with the interior penalty method for the advection-diffusion-reaction equation.

```
template <int dim>
void cdGMethod<dim>::assemble_system(
                                hp::DoFHandler<dim> &dof_handler)
{
```

The `UpdateFlags` manage which parts of the test functions need to be recalculated when moving to a new cell or face. For instance here we do not need to know the Hessian of the test functions, so we simply do not calculate it. The flag `update_JxW_values` refers to the Jacobian times the weight at each quadrature point.

```
  const UpdateFlags update_flags = update_values
                                   | update_gradients
                                   | update_quadrature_points
                                   | update_JxW_values;
  const UpdateFlags face_update_flags = update_values
                                   | update_quadrature_points
                                   | update_JxW_values
                                   | update_normal_vectors
                                   | update_gradients;
  const UpdateFlags nbr_face_update_flags = update_values
                                   | update_gradients;
```

The **`FEValues`** and **`FEFaceValues`** classes give an interface to the shape functions on a cell or face. We need only one for the values on the cells which will take continuous or discontinuous shape functions automatically depending on the `active_fe_index` on the cell. The **`FEFaceValues`** are only required on discontinuous cells (and their neighbours, denoted `nbr`). Finally on the interface we will always assemble from a discontinuous cell to a continuous one, so we have a special **`FEValues`** object for the continuous neighbour of a discontinuous cell.

```
  hp::FEValues<dim> hp_fe_v (fe_collection,
                             q_collection,
                             update_flags);
  FEFaceValues<dim> d_fe_face_v(d_fe,
```

```
                                face_quadrature,
                                face_update_flags);
FEFaceValues<dim> d_fe_face_v_nbr(d_fe,
                                  face_quadrature,
                                  nbr_face_update_flags);
FEValues<dim> c_fe_v_nbr (c_fe,
                          q_collection[CG_NOTHING],
                          update_flags);
```

On each cell we construct local matrices which will be then fed into the global `system_matrix`. In the most complicated case, that of two adjacent discontinuous cells, we need four matrices representing the interior of the cell (`i`) and the exterior (i.e., the neighbour, `e`) and the coupling between them.

```
FullMatrix<double> ui_vi_matrix;
FullMatrix<double> ue_vi_matrix;
FullMatrix<double> ui_ve_matrix;
FullMatrix<double> ue_ve_matrix;
Vector<double>     cell_vector;
std::vector<unsigned int> cell_dof_indices;
std::vector<unsigned int> nbr_dof_indices;
```

In order to ensure we pick up the correct element of **`FESystem`** on each cell, we use an extractor. Note that as we constructed each of `c_fe` and `d_fe` in the opposite order these match the enumeration of `fe_collection`.

```
const FEValuesExtractors::Scalar  continuous(CG_NOTHING);
const FEValuesExtractors::Scalar  discontinuous(NOTHING_DG);

typename hp::DoFHandler<dim>::active_cell_iterator
           cell = dof_handler.begin_active(),
           endc = dof_handler.end();
for ( ; cell!=endc; ++cell)
{
```

We visit each cell in turn and assemble either a discontinuous or continuous element.

```
  hp_fe_v.reinit(cell);
  const FEValues<dim> &fe_v = hp_fe_v.get_present_fe_values();

  ui_vi_matrix.reinit (cell->get_fe().dofs_per_cell,
                       cell->get_fe().dofs_per_cell);
  cell_vector.reinit(cell->get_fe().dofs_per_cell);
  cell_dof_indices.resize (cell->get_fe().dofs_per_cell);
  cell->get_dof_indices (cell_dof_indices);

  if(cell_is_c(cell))
    ad_equation.assemble_cell_term (fe_v,continuous,
                                    ui_vi_matrix,
```

```
                                        cell_vector);
    else
      ad_equation.assemble_cell_term (fe_v,discontinuous,
                                        ui_vi_matrix,
                                        cell_vector);
```

Calling `distribute_local_to_global` is more efficient than adding each local matrix and right hand side to the mass matrix and system right hand side, even though `constraints` is empty.

```
    constraints.distribute_local_to_global (ui_vi_matrix,
                                            cell_vector,
                                            cell_dof_indices,
                                            system_matrix,
                                            system_rhs);
```

Although we visit each cell once, if we were to loop over each face we would end up visiting each (except for the boundary) twice. At each face we therefore determine if the face is at the boundary, and if not we assemble only if the neighbour has a higher index. However if the neighbour is continuous we must assemble here as we do not ever consider the neighbours of continuous cells. This is more efficient than reformulating the interior penalty method (9.2.2) as a sum over cells and visiting each edge twice.

```
    if (cell_is_c(cell)) continue;
    for (unsigned int face_no=0;
          face_no<GeometryInfo<dim>::faces_per_cell;
          ++face_no)
    {
```

Case 1: Face at boundary.

```
      if(cell->face(face_no)->at_boundary())
      {
        d_fe_face_v.reinit(cell, face_no);
        ui_vi_matrix.reinit (cell->get_fe().dofs_per_cell,
                             cell->get_fe().dofs_per_cell);
        cell_vector.reinit(cell->get_fe().dofs_per_cell);
        ad_equation.assemble_boundary_term(d_fe_face_v,
                                           ui_vi_matrix,
                                           cell_vector);
        constraints.distribute_local_to_global(ui_vi_matrix,
                                               cell_vector,
                                               cell_dof_indices,
                                               system_matrix,
                                               system_rhs);
      }
```

```
      else
      {
```

Case 2: Neighbour is discontinuous and has higher index.

```
        typename hp::DoFHandler<dim>::cell_iterator nbr
                                    =cell->neighbor(face_no);
        if(cell_is_d(nbr) && (cell->index() < nbr->index()))
        {
          const unsigned int nbr_face_no
                          =cell->neighbor_of_neighbor(face_no);
          d_fe_face_v.reinit(cell, face_no);
          d_fe_face_v_nbr.reinit(nbr, nbr_face_no);
          unsigned int cell_dofs = cell->get_fe().dofs_per_cell;
          unsigned int nbr_dofs = nbr->get_fe().dofs_per_cell;
```

We do not allow different polynomial degrees on neighbouring cells. This is not possible with the current code but we add an `Assert` here to remind us if the code is altered in future. The extension to variable polynomial degree is possible using existing `deal.ii` capabilities.

```
          Assert(cell_dofs==nbr_dofs,
                  ExcMessage("Cell dofs must match nbr dofs"));

          ui_vi_matrix.reinit (cell_dofs,cell_dofs);
          ui_ve_matrix.reinit (cell_dofs,nbr_dofs);
          ue_vi_matrix.reinit (nbr_dofs,cell_dofs);
          ue_ve_matrix.reinit (nbr_dofs,nbr_dofs);
          ad_equation.assemble_face_term (d_fe_face_v,
                                          d_fe_face_v_nbr,
                                          ui_vi_matrix,
                                          ue_vi_matrix,
                                          ui_ve_matrix,
                                          ue_ve_matrix);

          nbr_dof_indices.resize (nbr_dofs);
          nbr->get_dof_indices (nbr_dof_indices);
          constraints.distribute_local_to_global(ui_vi_matrix,
                                                 cell_dof_indices,
                                                 system_matrix);
          constraints.distribute_local_to_global(ui_ve_matrix,
                                                 nbr_dof_indices,
                                                 cell_dof_indices,
                                                 system_matrix);
          constraints.distribute_local_to_global(ue_vi_matrix,
                                                 cell_dof_indices,
                                                 nbr_dof_indices,
                                                 system_matrix);
          constraints.distribute_local_to_global(ue_ve_matrix,
```

```
                                                nbr_dof_indices,
                                                system_matrix);
        }
```

Case 3: Neighbour is continuous.

```
        else if(cell_is_c(nbr))
        {
          c_fe_v_nbr.reinit(nbr);
          d_fe_face_v.reinit(cell, face_no);
          unsigned int cell_dofs = cell->get_fe().dofs_per_cell;
          unsigned int nbr_dofs = nbr->get_fe().dofs_per_cell;

          ui_vi_matrix.reinit (cell_dofs,cell_dofs);
          ui_ve_matrix.reinit (cell_dofs,nbr_dofs);
          ue_vi_matrix.reinit (nbr_dofs,cell_dofs);
          ue_ve_matrix.reinit (nbr_dofs,nbr_dofs);

          ad_equation.assemble_interface_term(d_fe_face_v,
                                              c_fe_v_nbr,
                                              ui_vi_matrix,
                                              ue_vi_matrix,
                                              ui_ve_matrix,
                                              ue_ve_matrix);
          nbr_dof_indices.resize (nbr_dofs);
          nbr->get_dof_indices (nbr_dof_indices);
          constraints.distribute_local_to_global(ui_vi_matrix,
                                                 cell_dof_indices,
                                                 system_matrix);
          constraints.distribute_local_to_global(ui_ve_matrix,
                                                 nbr_dof_indices,
                                                 cell_dof_indices,
                                                 system_matrix);
          constraints.distribute_local_to_global(ue_vi_matrix,
                                                 cell_dof_indices,
                                                 nbr_dof_indices,
                                                 system_matrix);
          constraints.distribute_local_to_global(ue_ve_matrix,
                                                 nbr_dof_indices,
                                                 system_matrix);
        }
      }
    }
  }
```

Finally we apply the boundary values to only the continuous part of the domain. Fortunately `deal.ii` has the capability to interpolate then apply `boundary_values`, a **`ParsedFunction`** object, to the `system_matrix` and solution.

```
  if(cdG_run)
```

```
  {
    std::map<unsigned int,double> bvs;
    VectorTools::interpolate_boundary_values (dof_handler,
                                              c_boundary_id,
                                              boundary_values,
                                              bvs);
    MatrixTools::apply_boundary_values ( bvs,
                                         system_matrix,
                                         cdG_solution,
                                         system_rhs);
  }
}
```

We use a direct solver to invert our matrix. `deal.ii` provides an interface with the library UMFPACK for this purpose, as well as many alternative numerical algorithms.

```
template <int dim>
void cdGMethod<dim>::solve(Vector<double> &solution)
{
  SparseDirectUMFPACK direct_solver;
  direct_solver.initialize (system_matrix);
  direct_solver.vmult (solution, system_rhs);
  constraints.distribute (solution);
}
```

Given an approximation we now have we wish to calculate the norms. This is not difficult using either **`VectorTools`** or our own code (one of the few cases of where this is advisable as the **`VectorTools`** norm calculations are very slow as they make no assumptions. We use `skipdGnorm` and `skipcdGnorm` in the parameter file to avoid calculating the norms and accelerate the computation for, e.g., testing). We therefore present a reduced example of the calculation showing the difference between the true solution and approximation in the L^2 norm, and the approximation jump norm.

```
template <int dim>
void cdGMethod<dim>::calculate_norms(
                                hp::DoFHandler<dim> &dof_handler,
                                Vector<double> &solution)
{
  Vector<double> L2norms;
```

We pick a higher quadrature formula here in order to better resolve the layer in the true solution.

```
  hp::QCollection<dim> q_collection_int;
  q_collection_int.push_back(QGauss<dim>(4));
  q_collection_int.push_back(QGauss<dim>(4));
```

The `MaskFunction` is required to switch between dG and cG regions. It is described in detail later.

```
  MaskFunction<dim> mask_function(2,dof_handler.get_tria());
  VectorTools::integrate_difference (dof_handler,
                                     solution,
                                     true_solution,
                                     L2norms,
                                     q_collection,
                                     VectorTools::L2_norm,
                                     &mask_function);
```

The `Vector` `L2norms` is a cellwise list of the L^2 norms. We now process it to a global norm $L^2(\Omega)$. The calculation of the jump norms is described with the `ADEquation` class.

```
  L2_norm = std::sqrt(L2norms.l2_norm());
  ad_equation.calculate_L2_jump_norms(dof_handler,
                        q_collection_int,
                        mapping_collection,
                        face_quadrature,
                        solution,
                        false,
                        L2_jump_norm,
                        L2norms);
}
```

We now output the solution in .vtk format, as well as any norms we have calculated. As we have two `FESystem` objects in `fe_collection` we get two solutions which gives the distinctive appearance as described in Section 9.1.

```
template <int dim>
void cdGMethod<dim>::output_results(
                                hp::DoFHandler<dim> &dof_handler,
                                Vector<double> &solution)
{
  std::string filename;
  std::string extension;
  if(!cdG_run) extension = "dG";
  else extension = "cdG";
  std::string reftext
              = boost::lexical_cast<std::string>( refinement );

  std::vector<std::string> solution_names;
  solution_names.push_back ("uc");
```

```
  solution_names.push_back ("ud");
  std::vector<
    DataComponentInterpretation::DataComponentInterpretation>
      data_component_interpretation;
  data_component_interpretation.push_back
      (DataComponentInterpretation::component_is_scalar);
  data_component_interpretation.push_back
      (DataComponentInterpretation::component_is_scalar);

  DataOut<dim,hp::DoFHandler<dim> > data_out_vtk;
  data_out_vtk.attach_dof_handler (dof_handler);
  data_out_vtk.add_data_vector (solution, solution_names,
               DataOut<dim,hp::DoFHandler<dim> >::type_dof_data,
               data_component_interpretation);
  data_out_vtk.build_patches ();

  filename = "solution"+extension+"ref"+reftext+".vtk";
  std::cout << "   Writing solution to <"
            << filename << ">..."<< std::endl;
  std::ofstream solution_output (filename.c_str());
  data_out_vtk.write_vtk (solution_output);
  solution_output.close();
```

We also output the grid for cdG approximations with continuous edges coloured.

```
  if(cdG_run)
  {
    filename = "grid"+extension+"ref"+reftext+".eps";
    std::cout << "   Writing grid to <" << filename
              << ">..." << std::endl;
    std::ofstream eps_grid_output (filename.c_str());

    GridOut grid_out;
    GridOutFlags::Eps<dim> eps_grid_flags;
      eps_grid_flags.color_lines_on_user_flag = true;
      eps_grid_flags.write_cell_numbers = false;
      eps_grid_flags.write_cell_number_level = false;
    grid_out.set_flags (eps_grid_flags);
    grid_out.write_eps (triangulation, eps_grid_output);
    eps_grid_output.close();
  }
```

Finally output a table with any calculated norms, the number of degrees of freedom used and the time taken to assemble and solve the problem.

```
  TableHandler table;

  table.add_value ("Norm", std::string("$L^2$"));
  table.add_value ("Value", L2_norm);
  table.add_value ("Norm", std::string("$L^2$ jump"));
```

```
  table.add_value ("Value", std::sqrt(L2_jump_norm.l1_norm()));
  double time;
  if(!cdG_run) time = dG_time;
  else time = cdG_time;
  table.add_value ("Norm", std::string("timer"));
  table.add_value ("Value", time);
  table.add_value ("Norm", std::string("dofs"));
  table.add_value ("Value", dof_handler.n_dofs());
  table.set_precision("Value", 4);
  table.set_scientific("Value", true);

  filename = "table-"+extension+"ref"+reftext+".txt";
  std::cout << "Writing norms to <"
            << filename << ">..." << std::endl;

  std::ofstream table_output_text(filename.c_str());
  table.write_text(table_output_text);
  table_output_text.close();
}
```

After each dG run we reinitialize the `system_matrix`, `sparsity_pattern`, `constraints` and `system_rhs` to be used again. Note that we do not destroy the `dof_handler` as we will require this to compare the dG and cdG approximations (for instance using an alternative implementation of `integrate_difference`).

```
template <int dim>
void cdGMethod<dim>::reset_system ()
{
  constraints.clear();
  sparsity_pattern.reinit(0,0,0);
  system_matrix.clear ();
  system_rhs.reinit(0);
}
```

The following routine manages the construction and solution for the approximations. As previously mentioned we always calculate both a discontinuous and continuous solution. This is inefficient but no suitable format exists to save and load an approximation including the **`DoFHandler`**.

```
template <int dim>
void cdGMethod<dim>::run ()
{
```

The **`Timer`** class allows us to measure time elapsed and simply interfaces with the system clock of most unix systems.

```
  Timer timer;
```

```
  run_number=0;
  cdG_run = false;
```

The grid need only be constructed once, but modified each time. The first time it will automatically be set up for the dG approximation.

```
  make_grid ();
  modify_grid ();
  setup_dofs (dG_dof_handler, dG_solution);

  timer.start();
  assemble_system (dG_dof_handler);
  solve (dG_solution);
  timer.stop();
  std::cout << "Time for assemble and solve "
            << timer() << "s" << std::endl;
  dG_time = timer();
  std::cout << std::endl;
  std::cout << "Calculating norms..." << std::endl;
  prm.enter_subsection ("Run Options");
    const bool skipdGnorm = prm.get_bool("Fast");
    const bool skipcdGnorm = prm.get_bool("skipcdGNorm");
  prm.leave_subsection ();
  if(!skipdGnorm) calculate_norms (dG_dof_handler, dG_solution);
  else std::cout << "      skipping calculate as Fast==true..."
                 <<std::endl;
  std::cout << "Output of results..." << std::endl;
  output_results (dG_dof_handler, dG_solution);
```

This completes the calculation of the dG approximation. We reset the system and advance the counters for the cdG approximation.

```
  ++run_number;
  cdG_run = true;
  std::cout << std::endl;
  reset_system ();
```

As now `cdG_run` is **`true`** calling `modify_grid` prepares for a cdG approximation based on the dG approximation.

```
  modify_grid ();
  setup_dofs (cdG_dof_handler, cdG_solution);

  timer.restart();
  assemble_system (cdG_dof_handler);
  solve (cdG_solution);
  timer.stop ();
  std::cout << "Time for assemble and solve "
            << timer() << "s" << std::endl;
  cdG_time = timer();
```

```
  std::cout << std::endl;
  std::cout << "Calculating norms..." << std::endl;
  if(!skipcdGnorm) calculate_norms(cdG_dof_handler,
                                   cdG_solution);
  else
    std::cout << "skipping calculate as skipcdGNorm==true..."
              << std::endl;
  std::cout << "Output of results..." << std::endl;
  output_results (cdG_dof_handler, cdG_solution);
}
```

The `main` routine simply reads the parameter file and passes the number of refinement steps to attempt into `run`. The call to `deallog.depth_console(0)` suppresses any internal `deal.ii` messages.

```
int main ()
{
  try
  {
    deallog.depth_console (0);

    ParameterHandler  prm;
    ParameterReader   param(prm);
    param.read_parameters("data.in");

    prm.enter_subsection("Run Options");
      const unsigned int ref_steps
          = prm.get_integer("refinement steps");
    prm.leave_subsection();

    for(unsigned int i=0;i<ref_steps;++i)
    {
      std::cout << "*****************************" <<std::endl;
      std::cout << std::endl;
      cdGMethod<DEAL_II_DIMENSION> cdg_method(prm,i);
      cdg_method.run ();
      std::cout << std::endl;
    }
  }
  catch (std::exception &exc)
  {
    std::cerr << std::endl << std::endl
              << "----------------------------------------------"
              << std::endl;
    std::cerr << "Exception on processing: " << std::endl
              << exc.what() << std::endl
              << "Aborting!" << std::endl
              << "----------------------------------------------"
              << std::endl;
```

```
    return 1;
  }
  catch (...)
  {
    std::cerr << std::endl << std::endl
              << "----------------------------------------------"
              << std::endl;
    std::cerr << "Unknown exception!" << std::endl
              << "Aborting!" << std::endl
              << "----------------------------------------------"
              << std::endl;
    return 1;
  }
  return 0;
}
```

Class: ADEquation

The class `ADEquation` describes the interior penalty method for the advection-diffusion-reaction equation (previous versions had no reaction, hence only AD) including any coefficients. It also includes the calculation of the jumps of an approximation, although for a more complicated example any norms should appear in a separate class. Here the assembly and norms are declared at **public** scope so we can access them from `cdGMethod`.

```
template <int dim>
class ADEquation
{
  public:
    ADEquation (ParameterHandler &param);
    ~ADEquation() {};

    //...Function prototypes for assembly omitted...
    //...Function prototypes for norms omitted...
```

The coefficients are at **private** scope as we can take them locally from the parameter file passed to the constructor.

```
  private:
    ParameterHandler               &prm;
    Functions::ParsedFunction<dim> rhs_function;
    Functions::ParsedFunction<dim> boundary_function;
    Functions::ParsedFunction<dim> beta_function;
    Functions::ParsedFunction<dim> divbeta_function;
    Functions::ParsedFunction<dim> reaction_function;
```

```
    double                          epsilon,sigma;
    signed int                      theta;
};
```

The main task of the constructor is to parse each function and make it available to the class. Performing the task in this way is inefficient (as we could pass the already parsed functions from the `cdGMethod` object) but it makes it easier to change to other discretizations if the parameters are stored within the class using them.

```
template <int dim>
ADEquation<dim>::ADEquation (ParameterHandler &param)
  :
  prm(param),
  rhs_function(1),
  boundary_function(dim),
  beta_function(dim),
  divbeta_function(1)
{
  prm.enter_subsection("Beta Data");
    beta_function.parse_parameters(prm);
  prm.leave_subsection();
  prm.enter_subsection("divBeta Data");
    divbeta_function.parse_parameters(prm);
  prm.leave_subsection();
  prm.enter_subsection("Boundary Data");
    boundary_function.parse_parameters(prm);
  prm.leave_subsection();
  prm.enter_subsection("Right Hand Side Data");
    rhs_function.parse_parameters(prm);
  prm.leave_subsection();
  prm.enter_subsection("Reaction Data");
    reaction_function.parse_parameters(prm);
  prm.leave_subsection();
  prm.enter_subsection("Equation Data");
    epsilon = prm.get_double("epsilon");
    theta = prm.get_integer("theta");
    sigma = prm.get_double("sigma");
  prm.leave_subsection();
}
```

The assemble terms are called from `cdGMethod::assemble_system` and so only assemble on one cell or face. For the cell terms the switch between continuous and discontinuous assembly is controlled by the passed values of `fe_v` and the extractor. If the two do not match an exception will be thrown when accessing, e.g., `fe_v[extractor].gradient`. The **`FEValues`** object is already initialized with the quadrature points on the cell (and other values as determined by `UpdateFlags` in

cdGMethod::assemble_system).

```
template <int dim>
void ADEquation<dim>::assemble_cell_term(
                    const FEValues<dim>& fe_v,
                    const FEValuesExtractors::Scalar extractor,
                    FullMatrix<double> &ui_vi_matrix,
                    Vector<double> &cell_vector) const
{
```

It makes for readable code (but is slightly less efficient) if we populate vectors with the values of the coefficients and Jacobians at each quadrature point.

```
  const std::vector<double> &JxW_vec = fe_v.get_JxW_values ();
  std::vector<Vector<double> > beta_vec(
                 fe_v.n_quadrature_points,Vector<double>(dim));
  std::vector<double> divbeta_vec (fe_v.n_quadrature_points);
  std::vector<double> reaction_vec (fe_v.n_quadrature_points);
  std::vector<double> rhs_vec (fe_v.n_quadrature_points);
```

The **ParsedFunction** inherits vector_value_list from the **Function** class. This is more efficient than calling vector_value for each quadrature point.

```
  beta_function.vector_value_list (fe_v.get_quadrature_points(),
                                   beta_vec);
  divbeta_function.value_list(fe_v.get_quadrature_points(),
                                   divbeta_vec);
  reaction_function.value_list(fe_v.get_quadrature_points(),
                                   reaction_vec);
  rhs_function.value_list (fe_v.get_quadrature_points(),
                                   rhs_vec);
```

We now loop over each quadrature point on the cell.

```
  for (unsigned int point=0;
       point<fe_v.n_quadrature_points;
       ++point)
  {
    const double JxW = JxW_vec[point];
```

We copy beta from a std::vector to a **Point**. This is because **ParsedFunction** does not support **Point** but it is easier to work with, in particular as * is overloaded correctly.

```
    Point<dim> beta;
    for(unsigned int i=0;i<dim;++i) beta(i) =beta_vec[point](i);
    const double divbeta = divbeta_vec[point];
    const double reaction = reaction_vec[point];
    const double rhs = rhs_vec[point];
```

We now loop over each degree of freedom and assemble its contribution for this quadrature point...

```
    for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
    {
      for (unsigned int j=0; j<fe_v.dofs_per_cell; ++j)
      {
```

...first for the diffusion term $\int_E \varepsilon \nabla_h u \cdot \nabla_h v \, \mathrm{d}\boldsymbol{x}$...

```
        ui_vi_matrix(i,j) +=
                  epsilon*fe_v[extractor].gradient(i,point)*
                  fe_v[extractor].gradient(j,point)*JxW;
```

...then the advection terms $\int_E -(b \cdot \nabla_h v)u - (\nabla_h \cdot b)uv \, \mathrm{d}\boldsymbol{x}$...

```
        ui_vi_matrix(i,j) -=
                  (beta*fe_v[extractor].gradient(i,point))*
                  fe_v[extractor].value(j,point) * JxW;
        ui_vi_matrix(i,j) -=
                  divbeta*fe_v[extractor].value(j,point)*
                  fe_v[extractor].value(i,point)* JxW;
```

...and finally the reaction term $\int_E cuv \, \mathrm{d}\boldsymbol{x}$ and right hand side $\int_E fv \, \mathrm{d}\boldsymbol{x}$.

```
        ui_vi_matrix(i,j) +=
                  reaction*fe_v[extractor].value(j,point)*
                  fe_v[extractor].value(i,point)* JxW;
      }
      cell_vector(i) += rhs *fe_v[extractor].value(i,point)*JxW;
    }
  }
}
```

Now we assemble faces lying in $\Gamma_{\mathrm{dG}} \setminus J$. Faces lying on the continuous boundary are covered by the boundary conditions. Even though we are now on a face, the construction is the same as on a cell. This is because **FEFaceValues** and **FEValues** are both inherit from the general class FEValuesBase.

```
template <int dim>
void ADEquation<dim>::assemble_boundary_term(
                              const FEFaceValues<dim>& fe_v,
                              FullMatrix<double> & ui_vi_matrix,
                              Vector<double> & cell_vector) const
{
  const std::vector<double> &JxW_vec = fe_v.get_JxW_values ();
  const std::vector<Point<dim> > &normals
                                    = fe_v.get_normal_vectors ();
  std::vector<Vector<double> > beta_vec (
                fe_v.n_quadrature_points, Vector<double>(dim));
```

```
std::vector<Vector<double> > g_vec(fe_v.n_quadrature_points,
                                   Vector<double>(dim));
beta_function.vector_value_list (fe_v.get_quadrature_points(),
                                 beta_vec);
boundary_function.vector_value_list
                        (fe_v.get_quadrature_points(), g_vec);

const double h
    = std::sqrt(std::pow(fe_v.get_cell()->diameter(),2.)/2.);
double esbyh = (epsilon*sigma)/h;
```

The cell must be discontinuous as we do not assemble boundary terms for continuous cells. As `NOTHING_DG` is not at this scope we have to use 1.

```
const FEValuesExtractors::Scalar  discontinuous(1);
for (unsigned int point=0;
      point<fe_v.n_quadrature_points;
      ++point)
{
  const double JxW = JxW_vec[point];
  const Point<dim> n = normals[point];
  const double g = g_vec[point](0);
  Point<dim> beta;
  for(unsigned int i=0;i<dim;++i) beta(i) =beta_vec[point](i);
```

First the diffusion terms $\int_e -\varepsilon\{\!\{\nabla_h u\}\!\}\cdot[\![v]\!] - \vartheta\varepsilon\{\!\{\nabla_h v\}\!\}\cdot[\![u]\!] + \varepsilon\sigma/h[\![u]\!]\cdot[\![v]\!]\,\mathrm{d}s\ldots$

```
  for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
  {
    for (unsigned int j=0; j<fe_v.dofs_per_cell; ++j)
    {
      ui_vi_matrix(i,j) -= epsilon*
                    fe_v[discontinuous].gradient(j,point)*
                    (fe_v[discontinuous].value(i,point)*n)*
                    JxW;
      ui_vi_matrix(i,j) -= theta*epsilon*
                    (fe_v[discontinuous].value(j,point)*n)*
                    fe_v[discontinuous].gradient(i,point)*
                    JxW;
      ui_vi_matrix(i,j) += esbyh*
                    fe_v[discontinuous].value(j,point)*
                    fe_v[discontinuous].value(i,point)*
                    JxW;
  }
```

$\ldots$ and the right hand side terms $\int_e \vartheta\varepsilon\nabla_h v g + \varepsilon\sigma/h v g\,\mathrm{d}s$.

```
    cell_vector(i) -= theta*epsilon*
                      fe_v[discontinuous].gradient(i,point)*n*
                      g*JxW;
    cell_vector(i) += esbyh*
```

```
                          fe_v[discontinuous].value(i,point)*g
                          *JxW;
    }
```

The advection terms are a little trickier as we have a different assembly for inflow and outflow boundaries.

```
    const double beta_n=beta * n;
    if (beta_n>0)
      for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
        for (unsigned int j=0; j<fe_v.dofs_per_cell; ++j)
```

We assemble the outflow boundary $\int_e b \cdot nuv \, \mathrm{d}s$.

```
          ui_vi_matrix(i,j) += beta_n *
                fe_v[discontinuous].value(j,point) *
                fe_v[discontinuous].value(i,point) *
                JxW;
    else
      for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
```

We assemble the inflow right hand side $\int_e (b \cdot n) vg \, \mathrm{d}s$.

```
          cell_vector(i) -= beta_n *
                            g *
                            fe_v[discontinuous].value(i,point) *
                            JxW;
  }
}
```

For a dG face in $\mathcal{E}_h^o \setminus J$ there are for each term four different assembly matrices corresponding to the coupling of the degrees of freedom on each cell and the coupling between the cells. If we were to visit each face twice, once from each side, we would only require two of these per visit.

```
template <int dim>
void
ADEquation<dim>::assemble_face_term(
                         const FEFaceValuesBase<dim>& fe_v,
                         const FEFaceValuesBase<dim>& fe_v_nbr,
                         FullMatrix<double> &ui_vi_matrix,
                         FullMatrix<double> &ue_vi_matrix,
                         FullMatrix<double> &ui_ve_matrix,
                         FullMatrix<double> &ue_ve_matrix) const
{
  const std::vector<double> &JxW_vec = fe_v.get_JxW_values ();
  const std::vector<Point<dim> > &normals
                                    = fe_v.get_normal_vectors ();
  std::vector<Vector<double> > beta_vec (
                  fe_v.n_quadrature_points,Vector<double>(dim));
```

```
beta_function.vector_value_list (fe_v.get_quadrature_points(),
                                 beta_vec);
const double h
     = std::sqrt(std::pow(fe_v.get_cell()->diameter(),2.)/2.);
double esbyh = (epsilon*sigma)/h;
const FEValuesExtractors::Scalar  discontinuous(1);
for (unsigned int point=0;
      point<fe_v.n_quadrature_points;
      ++point)
{
  const double JxW = JxW_vec[point];
  const Point<dim> n = normals[point];
  Point<dim> beta;
  for(unsigned int i=0;i<dim;++i) beta(i) =beta_vec[point](i);
  for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
  {
    for (unsigned int j=0; j<fe_v.dofs_per_cell; ++j)
    {
```

First assemble the diffusion terms on each cell...

```
      ui_vi_matrix(i,j) -= 0.5*epsilon*
                   (fe_v[discontinuous].gradient(j,point)*n)*
                   fe_v[discontinuous].value(i,point)*
                   JxW;
      ui_vi_matrix(i,j) -= 0.5*epsilon*theta*
                   (fe_v[discontinuous].gradient(i,point)*n)*
                   fe_v[discontinuous].value(j,point)*
                   JxW;
      ui_vi_matrix(i,j) += esbyh*
                   fe_v[discontinuous].value(j,point)*
                   fe_v[discontinuous].value(i,point)*
                   JxW;
```

...then between this cell and the neighbour...

```
      ui_ve_matrix(i,j) += 0.5*epsilon*
                (fe_v[discontinuous].gradient(j,point)*n)*
                fe_v_nbr[discontinuous].value(i,point)*
                JxW;
      ui_ve_matrix(i,j) -= 0.5*epsilon*theta*
                (fe_v_nbr[discontinuous].gradient(i,point)*n)*
                fe_v[discontinuous].value(j,point)*
                JxW;
      ui_ve_matrix(i,j) -= esbyh*
                fe_v_nbr[discontinuous].value(i,point)*
                fe_v[discontinuous].value(j,point)*
                JxW;
```

...then between the neighbour and this cell...

```
        ue_vi_matrix(i,j) += 0.5*epsilon*theta*
                  (fe_v[discontinuous].gradient(i,point)*n)*
                  fe_v_nbr[discontinuous].value(j,point)*
                  JxW;
        ue_vi_matrix(i,j) -= 0.5*epsilon*
                  (fe_v_nbr[discontinuous].gradient(j,point)*n)*
                  fe_v[discontinuous].value(i,point)*
                  JxW;
        ue_vi_matrix(i,j) -= esbyh*
                  fe_v[discontinuous].value(i,point)*
                  fe_v_nbr[discontinuous].value(j,point)*
                  JxW;
```

...then on the neighbour cell.

```
        ue_ve_matrix(i,j) += 0.5*epsilon*
                  (fe_v_nbr[discontinuous].gradient(j,point)*n)*
                  fe_v_nbr[discontinuous].value(i,point)*
                  JxW;
        ue_ve_matrix(i,j) += 0.5*epsilon*theta*
                  (fe_v_nbr[discontinuous].gradient(i,point)*n)*
                  fe_v_nbr[discontinuous].value(j,point)*
                  JxW;
        ue_ve_matrix(i,j) += esbyh*
                  fe_v_nbr[discontinuous].value(i,point)*
                  fe_v_nbr[discontinuous].value(j,point)*
                  JxW;
      }
    }
```

Now we do the same for the advection terms with the added complication of the direction of flow.

```
    const double beta_n = beta*n;
    if (beta_n>0)
    {
      for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
      {
        for (unsigned int j=0; j<fe_v.dofs_per_cell; ++j)
        {
          ui_vi_matrix(i,j) += beta_n *
                               fe_v[discontinuous].value(j,point) *
                               fe_v[discontinuous].value(i,point) *
                               JxW;
          ui_ve_matrix(i,j) -= beta_n *
                           fe_v[discontinuous].value(j,point) *
                           fe_v_nbr[discontinuous].value(i,point) *
                           JxW;
        }
```

```
      }
    }
    else
    {
```

Note that the signs do not change as we have $b \cdot n^+$ not $b \cdot n^-$.

```
      for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
      {
        for (unsigned int j=0; j<fe_v_nbr.dofs_per_cell; ++j)
        {
          ue_vi_matrix(i,j) += beta_n *
                       fe_v_nbr[discontinuous].value(j,point) *
                       fe_v[discontinuous].value(i,point)*
                       JxW;
          ue_ve_matrix(i,j) -= beta_n *
                       fe_v_nbr[discontinuous].value(j,point) *
                       fe_v_nbr[discontinuous].value(i,point) *
                       JxW;
        }
      }
    }
  }
}
```

To assemble terms on the interface J requires a separate routine as the continuous **FEValues** object will return an exception if called on a face (or alternatively a **FEFaceValues** object cannot be initialized for a continuous element). We must therefore manually find the location of each quadrature point on the unit cell and extract the shape function values/gradients directly.

```
template <int dim>
void ADEquation<dim>::assemble_interface_term (
                        const FEFaceValuesBase<dim>& fe_v,
                        const FEValuesBase<dim>& fe_v_nbr,
                        FullMatrix<double> &ui_vi_matrix,
                        FullMatrix<double> &ue_vi_matrix,
                        FullMatrix<double> &ui_ve_matrix,
                        FullMatrix<double> &ue_ve_matrix) const
{
  const std::vector<double> &JxW_vec = fe_v.get_JxW_values ();
  const std::vector<Point<dim> > &normals
                                    = fe_v.get_normal_vectors ();
  std::vector<Vector<double> > beta_vec (
                  fe_v.n_quadrature_points,Vector<double>(dim));
```

Get the quadrature points in real space and transform them to the unit cell.

```
  std::vector<Point<dim> > q_points
```

```
                                = fe_v.get_quadrature_points();
for(unsigned int i=0;i<q_points.size();++i)
  q_points[i]
      = fe_v_nbr.get_mapping().transform_real_to_unit_cell
                          (fe_v_nbr.get_cell(),q_points[i]);

beta_function.vector_value_list (fe_v.get_quadrature_points(),
                                 beta_vec);
const double h
    = std::sqrt(std::pow(fe_v.get_cell()->diameter(),2.)/2.);
const double sigmabyh = sigma/h;
const FEValuesExtractors::Scalar  discontinuous(1);

for (unsigned int point=0;
      point<fe_v.n_quadrature_points;
      ++point)
{
  const double JxW = JxW_vec[point];
  const Point<dim> n = normals[point];
  const Point<dim> pt = q_points[point];
  Point<dim> beta;
  for(unsigned int i=0;i<dim;++i) beta(i) =beta_vec[point](i);
```

We still have four different routines for the various couplings inside and between cells. Note that on continuous elements `fe_v[discontinuous].value` is replaced by `fe_v.get_fe().shape_value`.

```
  for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
  {
    for (unsigned int j=0; j<fe_v.dofs_per_cell; ++j)
    {
      ui_vi_matrix(i,j) -= 0.5*epsilon*
                   (fe_v[discontinuous].gradient(j,point)*n)*
                   fe_v[discontinuous].value(i,point)*
                   JxW;
      ui_vi_matrix(i,j) -= 0.5*epsilon*theta*
                   (fe_v[discontinuous].gradient(i,point)*n)*
                   fe_v[discontinuous].value(j,point)*
                   JxW;
      ui_vi_matrix(i,j) += epsilon*sigmabyh*
                   fe_v[discontinuous].value(j,point)*
                   fe_v[discontinuous].value(i,point)*
                   JxW;
      ui_ve_matrix(i,j) += 0.5*epsilon*
                   (fe_v[discontinuous].gradient(j,point)*n)*
                   fe_v_nbr.get_fe().shape_value(i,pt)*
                   JxW;
      ui_ve_matrix(i,j) -= 0.5*epsilon*theta*
                   (fe_v_nbr.get_fe().shape_grad(i,pt)*n)*
```

```
                    fe_v[discontinuous].value(j,point)*
                    JxW;
      ui_ve_matrix(i,j) -= epsilon*sigmabyh*
                    fe_v_nbr.get_fe().shape_value(i,pt)*
                    fe_v[discontinuous].value(j,point)*
                    JxW;
      ue_vi_matrix(i,j) += 0.5*epsilon*theta*
                    (fe_v[discontinuous].gradient(i,point)*n)*
                    fe_v_nbr.get_fe().shape_value(j,pt)*
                    JxW;
      ue_vi_matrix(i,j) -= 0.5*epsilon*
                    (fe_v_nbr.get_fe().shape_grad(j,pt)*n)*
                    fe_v[discontinuous].value(i,point)*
                    JxW;
      ue_vi_matrix(i,j) -= epsilon*sigmabyh*
                    fe_v[discontinuous].value(i,point)*
                    fe_v_nbr.get_fe().shape_value(j,pt)*
                    JxW;
      ue_ve_matrix(i,j) += 0.5*epsilon*
                    (fe_v_nbr.get_fe().shape_grad(j,pt)*n)*
                    fe_v_nbr.get_fe().shape_value(i,pt)*
                    JxW;
      ue_ve_matrix(i,j) += 0.5*epsilon*theta*
                    (fe_v_nbr.get_fe().shape_grad(i,pt)*n)*
                    fe_v_nbr.get_fe().shape_value(j,pt)*
                    JxW;
      ue_ve_matrix(i,j) += epsilon*sigmabyh*
                    fe_v_nbr.get_fe().shape_value(i,pt)*
                    fe_v_nbr.get_fe().shape_value(j,pt)*
                    JxW;
    }
  }
```

For the advection terms we must again consider the direction of flow.

```
  const double beta_n = beta*n;
  if (beta_n>0)
  {
    for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
    {
      for (unsigned int j=0; j<fe_v.dofs_per_cell; ++j)
      {
        ui_vi_matrix(i,j) += beta_n *
                        fe_v[discontinuous].value(j,point) *
                        fe_v[discontinuous].value(i,point) *
                        JxW;
        ui_ve_matrix(i,j) -= beta_n *
                        fe_v[discontinuous].value(j,point) *
                        fe_v_nbr.get_fe().shape_value(i,pt) *
                        JxW;
```

```
        }
      }
    }
    else
    {
      for (unsigned int i=0; i<fe_v.dofs_per_cell; ++i)
      {
        for (unsigned int j=0; j<fe_v_nbr.dofs_per_cell; ++j)
        {
          ue_vi_matrix(i,j) += beta_n *
                               fe_v_nbr.get_fe().shape_value(j,pt) *
                               fe_v[discontinuous].value(i,point)*
                               JxW;
          ue_ve_matrix(i,j) -= beta_n *
                               fe_v_nbr.get_fe().shape_value(j,pt) *
                               fe_v_nbr.get_fe().shape_value(i,pt) *
                               JxW;
        }
      }
    }
  }
}
```

`Deal.ii` does not have a function to calculate the jumps of a discontinuous function. The following function performs this task. Extensions to weighted L^2 norms for, e.g., non constant b, are reasonably simple.

```
template <int dim>
void ADEquation<dim>::calculate_L2_jump_norms(
                hp::DoFHandler<dim> &dof_handler,
                hp::QCollection<dim>  &q_collection,
                hp::MappingCollection<dim> &mapping_collection,
                const QGauss<dim-1> &face_quadrature,
                const Vector<double> &solution,
                Vector<double> &norms) const
{
  norms.reinit (dof_handler.get_tria().n_faces(),0.);
  norms_with_sigma.reinit (dof_handler.get_tria().n_faces(),0.);
```

Each face has two values corresponding to the `CG_NOTHING` and `NOTHING_DG` components. We will have to make sure that we pick the correct one on each face.

```
  std::vector<Vector<double> > face_values;
  std::vector<Vector<double> > nbr_face_values;
  std::vector<double> g;
  std::vector<Point<dim> > normals;
  std::vector<double> weighting;
  std::vector<Vector<double> > beta_vec;
  double jumpterm = 0.;
```

The procedure is much like assembly (hence why it is included in this class). Now however we do not need the function gradients.

```
  const UpdateFlags cell_update_flags = update_values
                                      | update_quadrature_points
                                      | update_JxW_values;
  const UpdateFlags face_update_flags = update_values
                                      | update_quadrature_points
                                      | update_JxW_values
                                      | update_normal_vectors;
  const UpdateFlags nbr_face_update_flags = update_values
                                      | update_quadrature_points
                                      | update_JxW_values
                                      | update_normal_vectors;
  hp::FEValues<dim> hp_fe_v (dof_handler.get_fe(),
                             q_collection,
                             cell_update_flags);
  FEFaceValues<dim> d_fe_face_v(dof_handler.get_fe()[1],
                                face_quadrature,
                                face_update_flags);
  FEFaceValues<dim> d_fe_face_v_nbr(dof_handler.get_fe()[1],
                                    face_quadrature,
                                    nbr_face_update_flags);
  FEValues<dim> c_fe_v_nbr (dof_handler.get_fe()[0],
                            q_collection[0],
                            cell_update_flags);

  typename hp::DoFHandler<dim>::active_cell_iterator
    cell = dof_handler.begin_active(),
    endc = dof_handler.end();
  for (; cell!=endc; ++cell)
  {
    hp_fe_v.reinit (cell);
    const unsigned int active_index = cell->active_fe_index();
```

If the cell is continuous we will either catch it from its dG neighbour or, if its neighbour is continuous also, there will be no jump.

```
    if(active_index==0) continue;
    for (unsigned int face_no=0;
          face_no<GeometryInfo<dim>::faces_per_cell; ++face_no)
    {
      const unsigned int face_index
                                = cell->face(face_no)->index();
```

At the boundary the jump is between the boundary conditions and the value of the approximation.

```
      if(cell->at_boundary(face_no))
      {
        d_fe_face_v.reinit(cell,face_no);
        const std::vector<double> &JxW
                                  = d_fe_face_v.get_JxW_values ();
        face_values.resize (d_fe_face_v.n_quadrature_points,
                            Vector<double>(2));
        d_fe_face_v.get_function_values (solution, face_values);
        g.resize(d_fe_face_v.n_quadrature_points);
        boundary_function.value_list (
                        d_fe_face_v.get_quadrature_points(),g);

        for (unsigned int p=0;
              p<d_fe_face_v.n_quadrature_points;
              ++p)
        {
```

Here we know we have a discontinuous cell, so we can pick out the `NOTHING_DG` part (i.e., that part in position 1).

```
          jumpterm = std::pow(face_values[p](1)-g[p],2.0);
          norms(face_index) += JxW[p]*jumpterm;
        }
      }
      else
```

We are on a discontinuous cell and need to act differently if our neighbour is continuous or discontinuous.

```
      {
        typename hp::DoFHandler<dim>::cell_iterator nbr
                                      =cell->neighbor(face_no);
        const unsigned int nbr_active_index
                                      = nbr->active_fe_index();
        const unsigned int nbr_face_no
                          =cell->neighbor_of_neighbor(face_no);

        if(active_index==1 and nbr_active_index==1
                and nbr->index() > cell->index())
        {
```

The neighbour is also discontinuous.

```
          d_fe_face_v.reinit(cell,face_no);
          d_fe_face_v_nbr.reinit(nbr,nbr_face_no);
          const std::vector<double> &JxW
                              = d_fe_face_v.get_JxW_values ();
          face_values.resize(d_fe_face_v.n_quadrature_points,
                              Vector<double>(2));
          Assert(d_fe_face_v_nbr.n_quadrature_points
```

```
                ==d_fe_face_v.n_quadrature_points,
                ExcNotImplemented());
      nbr_face_values.resize
                  (d_fe_face_v_nbr.n_quadrature_points,
                  Vector<double>(2));
      d_fe_face_v.get_function_values(solution,face_values);
      d_fe_face_v_nbr.get_function_values(solution,
                                          nbr_face_values);

      for (unsigned int p=0;
            p<d_fe_face_v.n_quadrature_points;
            ++p)
      {
        jumpterm = std::pow(face_values[p](1),2.0)
                +std::pow(nbr_face_values[p](1),2.0)
                -2.*face_values[p](1)*nbr_face_values[p](1);
        norms(face_index) += JxW[p]*jumpterm;
      }
    }
    else if(active_index==1 and nbr_active_index==0)
    {
```

The neighbour cell is continuous. We need to pick out the values "by hand" as we did in `assemble_interface_term`.

```
      d_fe_face_v.reinit(cell,face_no);
      c_fe_v_nbr.reinit(nbr);
      const std::vector<double> &JxW
                          = d_fe_face_v.get_JxW_values ();
      face_values.resize(d_fe_face_v.n_quadrature_points,
                          Vector<double>(2));
      d_fe_face_v.get_function_values(solution,face_values);
      std::vector<Point<dim> > q_points
                        = d_fe_face_v.get_quadrature_points();
      for (unsigned int p=0;
            p<d_fe_face_v.n_quadrature_points;
            ++p)
      {
        VectorTools::point_value(mapping_collection,
                                 dof_handler,
                                 solution,
                                 q_points[p],
                                 ptval);
```

Unfortunately `VectorTools::point_value` may not return the correct value on the interface of continuous and discontinuous edges. We must therefore check which entry is non-zero.

```
        double nbr_face_value;
```

```
            if(ptval(0)==0) nbr_face_value = ptval(1);
            else nbr_face_value = ptval(0);

            jumpterm = std::pow(face_values[p](1),2.0)
                      +std::pow(nbr_face_value,2.0)
                      -2.*face_values[p](1)*nbr_face_value;
            norms(face_index) += JxW[p]*jumpterm;
          }
        }
```

The final logical combination is a discontinuous neighbour with a lower index, which we will visit when we are on the neighbouring cell, and so we do nothing.

```
        else {\\...empty... }
      }
    }
  }
}
```

Finally we have to instruct the compiler to instantiate the class in two dimensions as `dim` is used explicitly in the class.

```
template class ADEquation<2>;
```

Class: MaskFunction

This function acts as a component mask, hiding either component of an approximation from the integration, i.e., so comparison to the true solution is done with the cG part on the continuous region and the dG part on the discontinuous region.

The class declaration shows the inheritance of **`Function`**. Therefore we only need to overload the `vector_value` call.

```
template <int dim>
class MaskFunction : public Function<dim>
{
  public:
    MaskFunction (unsigned int components,
                  const Triangulation<dim> &tria);
    virtual void vector_value (const Point<dim>   &p,
                               Vector<double>     &value) const;
  private:
    Triangulation<dim> triangulation;
};
```

We must copy the triangulation as `vector_value` needs to know about it but it cannot be passed directly (as it must match the prototype in **`Function`**).

```
template <int dim>
MaskFunction<dim>::MaskFunction(unsigned int components,
                                const Triangulation<dim> &tria)
  :
  Function<dim> (components)
{
  triangulation.copy_triangulation(tria);
}
```

The `vector_value` function simply works out whether the passed point is on a continuous or discontinuous cell and sets value to be 1 or 0 respectively. When a `MaskFunction` object is passed into **`VectorTools`**`::integrate_difference` the value will be multiplied by the integration at each point.

```
template <int dim>
void MaskFunction<dim>::vector_value(const Point<dim> &p,
                                     Vector<double> &value) const
{
```

First find the active cell to which this point belongs.

```
  std::pair<typename Triangulation<dim>::active_cell_iterator,
            Point<dim> > cell;
  cell
   = GridTools::find_active_cell_around_point(MappingQ1<dim>(),
                                              triangulation,
                                              p);
  Assert(this->n_components==2,ExcNotImplemented());
  value.reinit(this->n_components);
  if(cell.first->material_id() == 'c')
  {
    value(0) = 1;
    value(1) = 0;
  }
  else if (cell.first->material_id() == 'd')
  {
    value(0) = 0;
    value(1) = 1;
  }
  else
  {
    std::cout << "Unknown material_id in MaskFunction"
              <<std::endl;
    Assert(false, ExcNotImplemented());
  }
}
```

As the dimension `dim` is used we also have to instruct the compiler to instantiate the two dimensional case.

```
template class MaskFunction<2>;
```

9.3 Parameter File

Here we include an example of a parameter file for Example 9.1.1. As explained previously `Boundary Data` and `True Solution` have to have two identical components to interface with `fe_collection`.

```
# Listing of Parameters
# ---------------------
subsection Beta Data
  set Function constants  = pi=3.141592
  set Function expression = 1;1
  set Variable names      = x,y         # default: x,y,t
end

subsection Boundary Data
  set Function constants  = e=1e-4
  set Function expression = 0;0
  set Variable names      = x,y    # default: x,y,t
end

subsection Equation Data
  # Diffusion coefficient
  set epsilon    = 1e-4 # default: 0.001
  # Penalty parameter
  set sigma      = 10.0
  # Switch between Interior Penalty types
  set theta      = -1   # default: 1
  # Domain maximum x
  set xmax       = 1.0
  # Domain minimum x
  set xmin       = 0.0
  # Domain maximum y
  set ymax       = 1.0
  # Domain minimum y
  set ymin       = 0.0
end

subsection Reaction Data
  set Function constants  =
  set Function expression = 0.0
  set Variable names      = x,y,t
end

subsection Right Hand Side Data
```

```
  set Function constants  = e=1e-4
  set Function expression = x+y-(exp(-1/e)-exp((x-1)/e))/(1-exp(-1/e))
     - (exp(-1/e)-exp((y-1)/e))/(1-exp(-1/e)) # default: 0
  set Variable names      = x,y   # default: x,y,t
end

subsection Run Options
  # If true do not calculate the dG norms
  set Fast                = false
  # L2 jump tolerance of dG solution
  set L2 jump tolerance   = 0.001   # default: 0.0001
  # Number of refinements of basic grid
  set initial refinement  = 4       # default: 3
  # print parameters at the start of the run?
  set print parameters    = false
  # Number of refinement iterations
  set refinement steps    = 1
  # If true we do not calculate the cdG norms
  set skipcdGNorm         = false
  # Is the true solution present?
  set true present        = true   # default: true
end

subsection True solution
  set Function constants  = e=1e-4
  set Function expression = (x-((exp(-1/e)-exp((x-1)/e))/(1-exp(-1/e)))
     )*(y- ((exp(-1/e)-exp((y-1)/e))/(1-exp(-1/e)))) ; (x-((exp(-1/e)-
     exp((x-1)/e))/(1-exp(-1/e))))*(y- ((exp(-1/e)-exp((y-1)/e))/(1-
     exp(-1/e)))) # default: 0
  set Variable names      = x,y  # default: x,y,t
end

subsection divBeta Data
  set Function constants  =
  set Function expression = 0
  set Variable names      = x,y # default: x,y,t
end

# ***************END OF PARAMETERS***************
```

Chapter 10

Summary

Here we summarise our analysis, paying particular attention to the objectives (O1)-(O3) as presented in Chapter 1, Section 1.2.

In Chapter 3 we presented a modified bilinear form which allowed us to show a stability result for the cdG method for singularly perturbed finite element problems. To do so we made several assumptions about the coefficients of the problem and splitting of the triangulation into continuous and discontinuous regions, in particular that the flow was of a minimum strength and strictly from $\mathcal{T}_{\text{cG}}$ to $\mathcal{T}_{\text{dG}}$. These assumptions were sufficient to demonstrate that stability can be achieved using considerably fewer degrees of freedom than required for approximations using the interior penalty dG finite element method, cf., (O1).

In Chapters 5 and 6 we studied the equations of incompressible miscible displacement, firstly in the time dependent case and then, for weighted spaces, in the stationary case. Reliable a posteriori error estimators were shown for the continuous time RT-dG finite element approximation (O2). As discussed, the regularity required to generate such an estimator is higher than the regularity that can be guaranteed in many industrial applications. We therefore continued our study of Objective (O2) by presenting abstract analysis for a posteriori error estimators for coupled problems. By simplifying the problem of interest to the stationary case, and simplifying our finite element method by using continuous elements to approximate both the pressure and concentration we showed how an a posteriori estimator could be constructed if both pressure and concentration belonged to $W^{1,\infty}(\Omega)$. This

motivated the discussion of weighted spaces. We showed that by making some (reasonable) assumptions on the solution of a stationary miscible displacement problem, and by using properties of the weighted spaces, we could outline an a posteriori error estimator in cases where the gradient of the pressure and concentration are not bounded. The extension of the application of weighted spaces to discontinuous methods and time dependent methods (for the coupled problem) remains open.

In Part III we considered the practical application of the continuous discontinuous Galerkin method, and in doing so extended the application to the equations of incompressible miscible displacement (O3). By locally super penalizing the jumps in the discontinuous method we showed convergence to the cdG method (or cG method by penalizing all jumps). This approach generalizes analysis already present in the literature and provided a convenient way to further investigate the cdG method. However a super penalized method has the same number of degrees of freedom as a dG method. We therefore in Chapter 8 demonstrated that without a priori knowledge of the solution to the advection diffusion reaction equation we can still achieve a reduction in the required number of degrees of freedom for a reasonable approximation. Extension of this approach to the IMD equations (as opposed to using the super penalty method) would be possible with some development with the `deal.ii` library.

The work in Chapter 8 suggests that the assumptions used in Chapter 3 are sufficient but not necessary to show stability. Generalisations of the stability proof, or an inf-sup result for the cdG method, have not been possible to achieve in available time, but the author believes that with more study more general results are achievable.

Glossary of Nomenclature

∇_h	Piecewise divergence operator. p.15
ρ	Where b and c are the advection and reaction coefficients respectively, $c - \frac{1}{2}\nabla \cdot b > \rho$ ..... (2.1.6)
$\mathcal{B}_\varepsilon$	Interior penalty bilinear form for the advection diffusion reaction equation... ..... (2.2.6)
$\mathcal{B}_d$	Diffusion parts of $\mathcal{B}_\varepsilon$ (similarly for IMD in (4.4.6)) ..... (2.2.3)
$\mathcal{B}_a$	Advection parts of $\mathcal{B}_\varepsilon$ ..... (2.2.4)
$\mathcal{B}_r$	Reaction parts of $\mathcal{B}_\varepsilon$ ..... (2.2.5)
$\mathcal{S}$	Penalty bilinear form ..... (7.1.2)
$\mathcal{B}_{cq}$	Bilinear form for convection, production and injection in IMD.. (7.3.2)
$\mathcal{B}_{cq}^{\text{alt}}$	Bilinear form for convection, production and injection in IMD, alternative form ..... (4.4.8)
$\{\!\{ \cdot \}\!\}$	Average of discontinuous function ..... (1.3.6)
$[\![\cdot]\!]$	Jump of discontinuous function ..... (1.3.6)
$\mathcal{C}l_h(\cdot)$	Clement interpolation operator ..... (5.1.7)
$\mathcal{O}s(\cdot)$	Oswald interpolation operator ..... (8.2.4)
$\Pi_h^{\mathcal{RT}}(\cdot)$	Raviart Thomas projection operator p.58
$\mathcal{SZ}_h(\cdot)$	Scott-Zhang projection operator p.32

$L^p(D)$	Lebesgue space order p on domain D ..... (1.3.1)
$W^{m,p}(D)$	Sobolev space: Functions whose weak derivatives up to order m are in $L^p(D)$ .... (1.3.2)
$H^m(D)$	Equivalent to $W^{m,2}(D)$ p.6
$\mathcal{K}^m_{\vec{a}}(\Omega)$	The weighted Sobolev (Babuška-Kondratiev) space. ..... (6.4.2)
Ω	Domain p.7
Ω_T	Domain in time, $\Omega_T := (0,T] \times \Omega$ p.7
$\mathcal{T}_h$	Triangulation ..... (1.3.6)
$\mathcal{E}_h$	Mesh skeleton ..... (1.3.6)
$\mathcal{E}^o_h$	Internal mesh skeleton ..... (1.3.6)
Γ	The union of elemental boundary edges p.7
h_E, h_e	Diameter of cell E or edge e p.7
h	Maximum h_E in $\mathcal{T}_h$ p.7
$\mathcal{T}_{\mathrm{cG}}, \mathcal{T}_{\mathrm{dG}}$	Triangulation on the continuous or discontinuous region p.22
J	The intersection of the continuous and discontinuous triangulations, $J := \overline{\mathcal{T}}_{\mathrm{cG}} \cap \overline{\mathcal{T}}_{\mathrm{dG}}$ p.22
$\mathcal{E}_{\mathrm{cG}}, \mathcal{E}_{\mathrm{dG}}$	The continuous and discontinuous Galerkin skeletons (note $\mathcal{E}_{\mathrm{cG}} := \mathcal{E}_h \setminus \mathcal{E}_{\mathrm{dG}}$ so J is only in $\mathcal{E}_{\mathrm{dG}}$) p.22
$\Gamma_{\mathrm{cG}}, \Gamma_{\mathrm{dG}}$	The intersection of Γ with $\overline{\mathcal{T}}_{\mathrm{cG}}$ or $\overline{\mathcal{T}}_{\mathrm{dG}}$ p.22
V_{cG}	Continuous piecewise polynomial space ..... (2.1.8)
V_{dG}	Discontinuous piecewise polynomial space ..... (2.2.2)
V_{cdG}	Continuous-Discontinuous piecewise polynomial space ..... (2.3.4)

Bibliography

[1] `Deal.II`*: A finite element differential equations analysis library.* http://www.dealii.org/.

[2] D. J. Acheson, *Elementary Fluid Dynamics*, Clarendon Press, Oxford, 3rd ed., 1990.

[3] R. A. Adams and J. J. Fournier, *Sobolev Spaces*, Elsevier, Oxford, 2nd ed., 2007.

[4] M. Ainsworth and J. Oden, *A unified approach to a posteriori error estimation using element residual methods*, Numer. Math., 65 (1993), pp. 23–50.

[5] M. Ainsworth and J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*, Pure and Applied Mathematics, Wiley-Interscience, 2000.

[6] B. Ammann and V. Nistor, *Weighted Sobolev spaces and regularity for polyhedral domains*, Comput. Method. Appl. M., 196 (2007), pp. 3650 – 3659.

[7] T. Apel and G. Lube, *Anisotropic mesh refinement in stabilized Galerkin methods*, Numer. Math., 74 (1996), pp. 261–282.

[8] D. N. Arnold, *An interior penalty finite element method with discontinuous elements*, SIAM J. Numer. Anal., 19 (1982), pp. 742–760.

[9] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM J. Numer. Anal., 39 (2001), pp. 1749–1779.

[10] B. Ayuso and L. D. Marini, *Discontinuous Galerkin methods for advection-diffusion-reaction problems*, SIAM J. Numer. Anal., 47 (2009), pp. 1391–1420.

[11] I. Babuska and M. Suri, *The hp version of the finite element method with quasiuniform meshes*, RAIRO - Modélisation mathématique et analyse numérique, 21 (1987), pp. 199–238.

[12] I. Babuška, *The finite element method with penalty*, Math. Comp., 72 (1973), pp. 221–228.

[13] C. Bacuta, V. Nistor, and L. T. Zikatanov, *Improving the rate of convergence of high-order finite elements on polyhedra ii: Mesh refinements and interpolation*, Numer. Func. Anal. Opt., 28 (2007), pp. 775–824.

[14] G. Baker, *Finite element methods for elliptic equations using nonconforming elements*, Math. Comp., 31 (1977), pp. 45–59.

[15] W. Bangerth, R. Hartmann, and G. Kanschat, `Deal.II` – *A general purpose object oriented finite element library*, ACM. T. Math. Software, 33 (2007), pp. 24/1–24/27.

[16] W. Bangerth and G. Kanschat, `Deal.II` *Differential Equations Analysis Library Technical Reference*. http://www.dealii.org/.

[17] R. E. Bank and A. Weiser, *Some a posteriori error estimators for elliptic partial differential equations*, Math. Comput., 44 (1985), pp. 283–301.

[18] C. Bardos and J. Rauch, *Maximal positive boundary value problems as limits of singular perturbation problems*, T. Am. Math. Soc., 270 (1982), pp. pp. 377–408.

[19] T. Barrios and R. Bustinza, *An a posteriori error analysis of an augmented discontinuous Galerkin formulation for Darcy flow*, Numer. Math., (2011), pp. 1–39. 10.1007/s00211-011-0410-3.

[20] S. Bartels, M. Jensen, and R. Müller, *Discontinuous Galerkin finite element convergence for incompressible miscible displacement problems of low regularity*, SIAM J. Numer. Anal., 47 (2009), pp. 3720–3743.

[21] F. Bassi and S. Rebay, *A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations*, J. Comp. Phys., 131 (1997), pp. 267 – 279.

[22] P. Bastian, *Numerical Computation of Multiphase Flows in Porous Media*, Habilitationsschrift, Christian-Albrechts-Universität Kiel, 1999.

[23] C. E. Baumann and J. T. Oden, *A discontinuous hp finite element method for convection-diffusion problems*, Comput. Method. Appl. M., 175 (1999), pp. 311 – 341.

[24] J. Bear, *Dynamics of Fluids in Porous Media*, Dover, New York, 1988.

[25] J. Bear and A. H.-D. Cheng, *Modeling Groundwater Flow and Contaminant Transport*, Theory and Applications of Transport in Porous Media, Springer, 2009.

[26] R. Becker, E. Burman, P. Hansbo, and M. G. Larson, *A reduced p^1-discontinuous Galerkin method*, Tech. Rep. 2003-13, Chalmers University of Technology, 2003.

[27] R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numerica, 10 (2001), pp. 1–102.

[28] M. Bertoldi, S. Fornaro, and L. Lorenzi, *Gradient estimates for parabolic problems with unbounded coefficients in non convex unbounded domains*, Forum Mathematicum, 19 (2007), pp. 603–632.

[29] M. Bertoldi and L. Lorenzi, *Estimates of the derivatives for parabolic operators with unbounded coefficients*, Trans. Amer. Math. Soc., 357 (2005), pp. 2627–2664.

[30] K. Bey and J. Tinsley Oden, *hp-version discontinuous Galerkin methods for hyperbolic conservation laws*, Comput. Method. Appl. M., 133 (1996), pp. 259–286.

[31] F. Bornemann, B. Erdmann, and R. Kornhuber, *A posteriori error estimates for elliptic problems in two and three space dimensions*, SIAM J. Numer. Anal., (1996), pp. 1188–1204.

[32] D. Braess, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University Press, Cambridge, 3rd ed., 2007.

[33] S. C. Brenner, *Poincaré–Friedrichs inequalities for piecewise H^1 functions*, SIAM J. Numer. Anal., 41 (2003), pp. 306–324.

[34] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer, New York, 3rd ed., 2008.

[35] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[36] F. Brezzi, D. Marini, and E. Süli, *Residual-free bubbles for advection-diffusion problems: The general error analysis*, Numer. Math., 85 (1998), pp. 31–47.

[37] F. Brezzi, L. D. Marini, and E. Süli, *Discontinuous Galerkin methods for first-order hyperbolic problems*, Math. Mod. Meth. Appl. S., 14 (2004), pp. 1893–1903.

[38] F. Brezzi and A. Russo, *Choosing bubbles for advection-diffusion problems*, Math. Mod. Meth. Appl. S., 4 (1994), pp. 571–587.

[39] A. N. Brooks and T. J. Hughes, *Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations*, Comput. Method. Appl. M., 32 (1982), pp. 199 – 259.

[40] R. Brown, *Some embeddings of weighted Sobolev spaces on finite measure and quasibounded domains*, J. Inequal. Appl., 2 (1998), pp. 325–356.

[41] A. Buffa, T. J. R. Hughes, and G. Sangalli, *Analysis of a multiscale discontinuous Galerkin method for convection-diffusion problems*, SIAM J. Numer. Anal., 44 (2006), pp. 1420–1440.

[42] E. Burman, A. Quarteroni, and B. Stamm, *Interior penalty continuous and discontinuous finite element approximations of hyperbolic equations*, J. Sci. Comp., 43 (2010), pp. 293–312. 10.1007/s10915-008-9232-6.

[43] E. Burman and P. Zunino, *A domain decomposition method based on weighted interior penalties for advection-diffusion-reaction problems*, SIAM J. Numer. Anal., 44 (2006), pp. 1612–1638.

[44] A. Cangiani, J. Chapman, E. Georgoulis, and M. Jensen, *On the stability of continuous discontinuous Galerkin finite element methods for singularly perturbed advection diffusion reaction equations.* In preparation.

[45] ——, *Implementation of the continuous-discontinuous Galerkin finite element method*, in Numerical Mathematics and Advanced Applications 2011, A. Cangiani, R. Davidchack, E. Georgoulis, A. Gorban, J. Levesley, and M. Tretyakov, eds., Springer, 2012.

[46] A. Cangiani, J. Chapman, E. H. Georgoulis, and M. Jensen, *On local super-penalization of interior penalty Galerkin methods*, Int. J. Numer. Anal. Model., (2012). Submitted.

[47] A. Cangiani, E. H. Georgoulis, and M. Jensen, *Continuous and discontinuous finite element methods for convection-diffusion problems: A comparison*, in International Conference on Boundary and Interior Layers, Göttingen, July 2006.

[48] C. Carstensen, *A posteriori error estimate for the mixed finite element method*, Math. Comp., 66 (1997), pp. 465–476.

[49] J. Chapman and M. Jensen, *Towards a posteriori error estimators for realistic problems in incompressible miscible displacement*, in Numerical Mathematics and Advanced Applications 2011, A. Cangiani, R. Davidchack, E. Georgoulis, A. Gorban, J. Levesley, and M. Tretyakov, eds., Springer, 2012.

[50] G. Chavent and J. Jaffre, *Mathematical Models and Finite Elements for Reservoir Simulation Single Phase, Multiphase and Multicomponent Flows through Porous Media*, vol. 17 of Studies in Mathematics and Its Applications, Elsevier, 1986.

[51] Y. Chen and W. Liu, *A posteriori error estimates of mixed methods for miscible displacement problems*, Int. J. Numer. Meth. Eng., 73 (2008), pp. 331–343.

[52] Z. Chen and R. Ewing, *Mathematical analysis for reservoir models*, SIAM J. Math. Anal., 30 (1999), pp. 431–453.

[53] P. Clment, *Approximation by finite element functions using local regularization*, Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique, 9 (1975), pp. 77–85.

[54] B. Cockburn, G. Karniadakis, and C. Shu, *The development of discontinuous Galerkin methods*, in Discontinuous Galerkin Methods: Theory, Computation, and Applications, B. Cockburn, G. Karniadakis, and C. Shu, eds., vol. 11 of Lecture Notes in Computational Science and Engineering, Springer, 2000.

[55] B. Cockburn and C.-W. Shu, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2440–2463.

[56] M. Cui, *A combined mixed and discontinuous Galerkin method for compressible miscible displacement problem in porous media*, J. Comput. Appl. Math., 198 (2007), pp. 19 – 34.

[57] ——, *Analysis of a semidiscrete discontinuous Galerkin scheme for compressible miscible displacement problem*, J. Comput. Appl. Math., 214 (2008), pp. 617–636.

[58] M. Dauge, *Elliptic Boundary Value Problems on Corner Domains: Smoothness and Asymptotics of Solutions*, vol. 1341 of Lecture Notes in Mathematics, Springer, 1988.

[59] C. Dawson and J. Proft, *Coupling of continuous and discontinuous Galerkin methods for transport problems*, Comput. Method. Appl. M., 191 (2002), pp. 3213 – 3231.

[60] C. Dawson, S. Sun, and M. F. Wheeler, *Compatible algorithms for coupled flow and transport*, Comput. Method. Appl. M., 193 (2004), pp. 2565 – 2580.

[61] P. R. B. Devloo, T. Forti, and S. M. Gomes, *A combined continuous-discontinuous finite element method for convection-diffusion problems*, Lat. Am. J. Solids Struct., 2 (2007), pp. 229–246.

[62] E. Doolan, J. J. H. Miller, and W. H. A. Schilders, *Uniform Numerical Methods for Problems with Initial and Boundary Layers*, Boole Press, 1980.

[63] J. Douglas and T. Dupont, *Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods*, in Lecture Notes in Physics, R. Glowinski and J. L. Lions, eds., vol. 58 of Lecture Notes in Physics, 1976, p. 207.

[64] J. Douglas, R. E. Ewing, and M. F. Wheeler, *The approximation of the pressure by a mixed method in the simulation of miscible displacement*, RAIRO Anal. Numer., 17 (1983), pp. 17–33.

[65] J. Douglas and J. E. Roberts., *Global estimates for mixed methods for second order elliptic equations*, Math. Comp., 44 (1985), pp. 39 – 52.

[66] R. G. Durán, *Mixed finite element methods*, in Mixed Finite Elements, Compatibility Conditions, and Applications, D. Boffi and L. Gastaldi, eds., vol. 1939 of Lecture Notes in Mathematics, Springer, 2008, pp. 1–44.

[67] A. Ern and J.-L. Guermond, *Theory and Practice of Finite Elements*, Springer-Verlag, New York, 2004.

[68] A. Ern, A. F. Stephansen, and P. Zunino, *A discontinuous Galerkin method with weighted averages for advection-diffusion equations with locally small and anisotropic diffusivity*, IMA J. Numer. Anal., 29 (2009), pp. 235–256.

[69] L. C. Evans, *Partial Differential Equations*, American Mathematical Society, Rhode Island, 2008.

[70] R. E. Ewing and M. F. Wheeler, *Galerkin methods for miscible displacement problems in porous media*, SIAM J. Numer. Anal., 17 (1980), pp. 351–365.

[71] X. B. Feng, *On existence and uniqueness results for a coupled system modeling miscible displacement in porous media*, J. Math. Anal. Appl., 194 (1995), pp. 883 – 910.

[72] S. Fornaro, G. Metafune, and E. Priola, *Gradient estimates for Dirichlet parabolic problems in unbounded domains*, J. Diff. Equ., 205 (2004), pp. 329 – 353.

[73] E. H. Georgoulis, *Discontinuous Galerkin Methods on Shape-Regular and Anisotropic Meshes*, PhD Thesis, Oxford Univeristy, 2003.

[74] E. H. Georgoulis and A. Lasis, *A note on the design of hp-version interior penalty discontinuous Galerkin finite element methods for degenerate problems*, IMA J. Numer. Anal., 26 (April 2006), pp. 381–390.

[75] E. H. Georgoulis and D. Loghin, *Norm preconditioners for discontinuous Galerkin hp-finite element methods*, SIAM J. Sci. Comp., 30 (2008), pp. 2447–2465.

[76] V. Gol'dshtein and A. Ukhlov, *Weighted Sobolev spaces and embedding theorems*, Trans. Amer. Math. Soc., 361 (2009), pp. 3829–3850.

[77] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Pitman Publishing, Massachusetts, 1985.

[78] J. Guzmán, *Local analysis of discontinuous Galerkin methods applied to singularly perturbed problems*, J. Numer. Math., 14 (2006), pp. 41–56.

[79] P. Hemker, *A singularly perturbed model problem for numerical computation*, J. Comput. Appl. Math., 76 (1996), pp. 277 – 285.

[80] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Springer, New York, 2008.

[81] P. Houston, D. Schötzau, and T. Wihler, *Energy norm a-posteriori error estimation of hp-adaptive discontinuous Galerkin methods for elliptic problems*, Math. Mod. Meth. Appl. S., 17 (2007), pp. 33–62.

[82] P. Houston, C. Schwab, and E. Süli, *Discontinuous hp-finite element methods for advection-diffusion-reaction problems*, SIAM J. Numer. Anal., 39 (2002), pp. 2133–2163.

[83] P. Houston, E. Süli, and T. P. Wihler, *A posteriori error analysis of hp-version discontinuous Galerkin finite-element methods for second-order quasi-linear elliptic PDEs*, IMA J. Numer. Anal., 28 (2008), pp. 245–273.

[84] T. Hughes and A. Brooks, *A multidimensional upwind scheme with no crosswind diffusion*, in Finite element methods for convection dominated flows, T. Hughes, ed., vol. 34 of Applied Mathematics Division, Göttingen, 1979, American Society of Mechanical Engineers.

[85] M. Jensen and R. Müller, *Stable Crank-Nicolson discretisation for incompressible miscible displacement problems of low regularity*, in Numerical Mathematics and Advanced Applications 2009, G. Kreiss, P. Lötstedt, A. Målqvist, and M. Neytcheva, eds., Springer Berlin Heidelberg, 2010, pp. 469–477.

[86] V. John and P. Knobloch, *On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations: Part I - A review*, Comput. Method. Appl. M., 196 (2007), pp. 2197 – 2215.

[87] ——, *On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations: Part II - Analysis for P_1 and Q_1 and finite elements*, Comput. Method. Appl. M., 197 (2008), pp. 1997 – 2014.

[88] C. Johnson and U. Nävert, *An analysis of some finite element methods for advection-diffusion problems*, in Analytical and Numerical Approaches to Asymptotic Problems in Analysis Proceedings of the Conference on Analytical and Numerical Approaches to Asymptotic Problems, O. Axelsson, L. Frank, and A. V. D. Sluis, eds., vol. 47 of North-Holland Mathematics Studies, North-Holland, 1981, pp. 99 – 116.

[89] O. A. Karakashian and F. Pascal, *A posteriori error estimates for a discontinuous Galerkin approximation of second-order elliptic problems*, SIAM J. Numer. Anal., 41 (2003), pp. 2374–2399.

[90] ——, *Convergence of adaptive discontinuous Galerkin approximations of second order elliptic problems*, SIAM J. Numer. Anal., 45 (2007), pp. 641–665.

[91] J. Kevorkian and J. Cole, *Multiple Scale and Singular Perturbation Methods*, vol. 114 of Applied Mathematical Sciences, Springer, 1996.

[92] K.-h. Kim, *A W_p^n-theory of parabolic equations with unbounded leading coefficients on non-smooth domains*, J. Math. Anal. Appl., 350 (2009), pp. 294–305.

[93] A. Kufner and A.-M. Sandig, *Some Applications of Weighted Sobolev Spaces*, B.G.Teubner GmbH, 1987.

[94] L. W. Lake, *Enhanced Oil Recovery*, Prentice Hall, 1996.

[95] M. G. Larson and A. Målqvist, *Goal oriented adaptivity for coupled flow and transport problems with applications in oil reservoir simulations*, Comp. Meth. Appl. Mech. Eng., 196 (2007), pp. 3546 – 3561.

[96] M. G. Larson and A. J. Niklasson, *Conservation properties for the continuous and discontinuous Galerkin methods*, Tech. Rep. 2000-08, Chalmers University of Technology, 2000.

[97] M. Latil, *Enhanced Oil Recovery*, Institut Français du Pétrole Publications, Éditions Technip, 1980.

[98] P. D. Lax and A. N. Milgram, *Parabolic equations*, in Contributions to the Theory of Partial Differential Equations, L. Bers, S. Bochner, and F. John, eds., vol. 33 of Annals of Mathematics Studies, Princeton University Press, 2000, p. 167190.

[99] P. Lesaint, *Finite element methods for symmetric hyperbolic equations*, Numer. Math., 21 (1973), pp. 244–255.

[100] H. Li, *Elliptic Equations with Singularities: A Priori Analysis and Numerical Approaches*, PhD Thesis, The Pennsylvania State University, 2008.

[101] H. Li, A. Mazzucato, and V. Nistor, *Analysis of the finite element method for transmission/mixed boundary value problems on general polygonal domains*, Elec. Trans. Numer. Anal., 37 (2010), pp. 41–69.

[102] G. M. Lieberman, *Second Order Parabolic Differential Equations*, World Scientific Publishing, Singapore, 1996.

[103] S. A. Lomov, *Introduction to the General Theory of Singular Perturbations*, vol. 112 of Translations of Mathematical Monographs, American Mathematical Society, 1993.

[104] J. M. Melenk, *hp-Finite Element Methods for Singular Perturbations*, vol. 1796 of Lecture Notes in Mathematics, Springer, 2003.

[105] A. Mikelić, *Mathematical theory of stationary miscible filtration*, J. Differ. Equations, 90 (1991), pp. 186 – 202.

[106] J. J. H. Miller, E. O'Riordan, and G. I. Shishkin, *Fitted Numerical Methods for Singular Perturbation Problems*, World Scientific Publishing Co., Inc., River Edge, NJ, 1996.

[107] K. Morton, *Numerical Solution of Convection-Diffusion Problems*, Chapman and Hall, London, 1996.

[108] H. Nguyen, M. Gunzburger, L. Ju, and J. Burkardt, *Adaptive anisotropic meshing for steady convection-dominated problems*, Comput. Method. Appl. M., 198 (2009), pp. 2964 – 2981.

[109] P. Oswald, *On a bpx-preconditioner for p1 elements*, Computing, 51 (1993), pp. 125–133. 10.1007/BF02243847.

[110] D. W. Peaceman, *Fundamentals of Numerical Reservoir Simulation*, Developments in Petroleum Science, Elsevier Scientific Publishing Company, 1977.

[111] I. Perugia and D. Schötzau, *On the coupling of local discontinuous Galerkin and conforming finite element methods*, J. Sci. Comp., 16 (2001), pp. 411–433.

[112] D. A. D. Pietro and A. Ern, *Mathematical Aspects of Discontinuous Galerkin Methods*, vol. 69 of Mathématiques et Applications, Springer, 2012.

[113] P. Raviart and J. Thomas, *A mixed finite element method for second order elliptic problems*, in Mathematical Aspects of the Finite Element Method, I. Galligani and E. Magenes, eds., vol. 606 of Lecture Notes in Mathematics, Springer, 1977.

[114] W. Reed and T. Hill, *Triangular mesh methods for the neutron transport equation*, Tech. Rep. LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM, 1973.

[115] B. Rivière, *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation*, SIAM, Philadelphia, 2009.

[116] B. Riviere, M. F. Wheeler, and V. Girault, *A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems*, SIAM J. Numer. Anal., 39 (2001), pp. 902–931.

[117] H.-G. Roos, M. Stynes, and L. Tobiska, *Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion and Flow Problems*, Springer-Verlag, Berlin, Second ed., 2008.

[118] F. Schieweck, *On the role of boundary conditions for CIP stabilization of higher order finite elements*, Elec. Trans. Numer. Anal., 32 (2008), pp. 1–16.

[119] C. Schwab, *p- and hp- Finite Element Methods: Theory and Applications in Solid Mechanics*, Clarendon Press, Oxford, 1998.

[120] C. Schwab and M. Suri, *The p and hp versions of the finite element method for problems with boundary layers*, Math. Comp., 65 (1996), pp. 1403 – 1429.

[121] L. R. Scott and S. Zhang, *Finite element interpolation of nonsmooth functions satisfying boundary conditions*, Math. Comp., 54 (1990), pp. 483–493.

[122] P. Solín and J. Ávila, *Equidistributed error mesh for problems with exponential boundary layers*, J. Comput. Appl. Math., 218 (2008), pp. 157 – 166.

[123] S. Sun, B. Rivière, and M. F. Wheeler, *A combined mixed finite element and discontinuous Galerkin method for miscible displacement problem in porous media.*, in Recent Progress in Computational and Applied PDEs, New York, 2002, Kluwer/Plenum, pp. 323–351.

[124] S. Sun and M. F. Wheeler, *Discontinuous Galerkin methods for coupled flow and reactive transport problems*, Appl. Numer. Math., 52 (2005), pp. 273 – 298.

[125] S. Sun and M. F. Wheeler, *A posteriori error estimation for discontinuous Galerkin approximations of reactive transport problems*, J. Sci. Comp., 22-23 (2005), pp. 501–530.

[126] S. Sun and M. F. Wheeler, *A posteriori error estimation and dynamic adaptivity for symmetric discontinuous Galerkin approximations of reactive transport problems*, Comput. Method. Appl. M., 195 (2006), pp. 632 – 652.

[127] M. F. Wheeler, *An elliptic collocation-finite element method with interior penalties*, SIAM J. Numer. Anal., 15 (1978), pp. 152–161.

[128] J. Yang, *A posteriori error of a discontinuous Galerkin scheme for compressible miscible displacement problems with molecular diffusion and dispersion*, Int. J. Numer. Meth. Fluids, 65 (2009), pp. 781–797.

[129] H. Zarin, *Continuous-discontinuous finite element method for convection-diffusion problems with characteristic layers*, J. Comput. Appl. Math., 231 (2009), pp. 626 – 636.